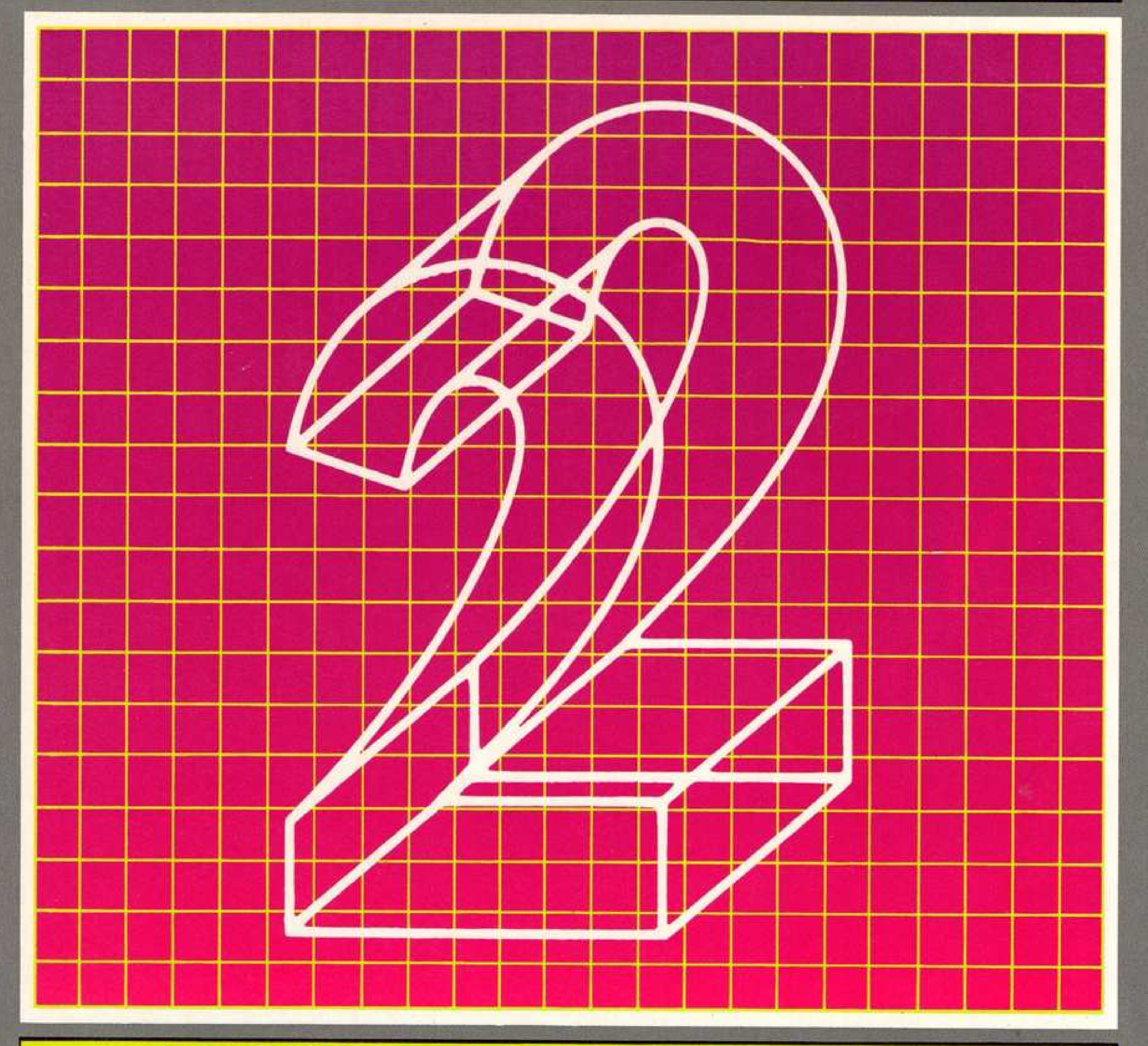
BIBLIOTECA PRACTICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

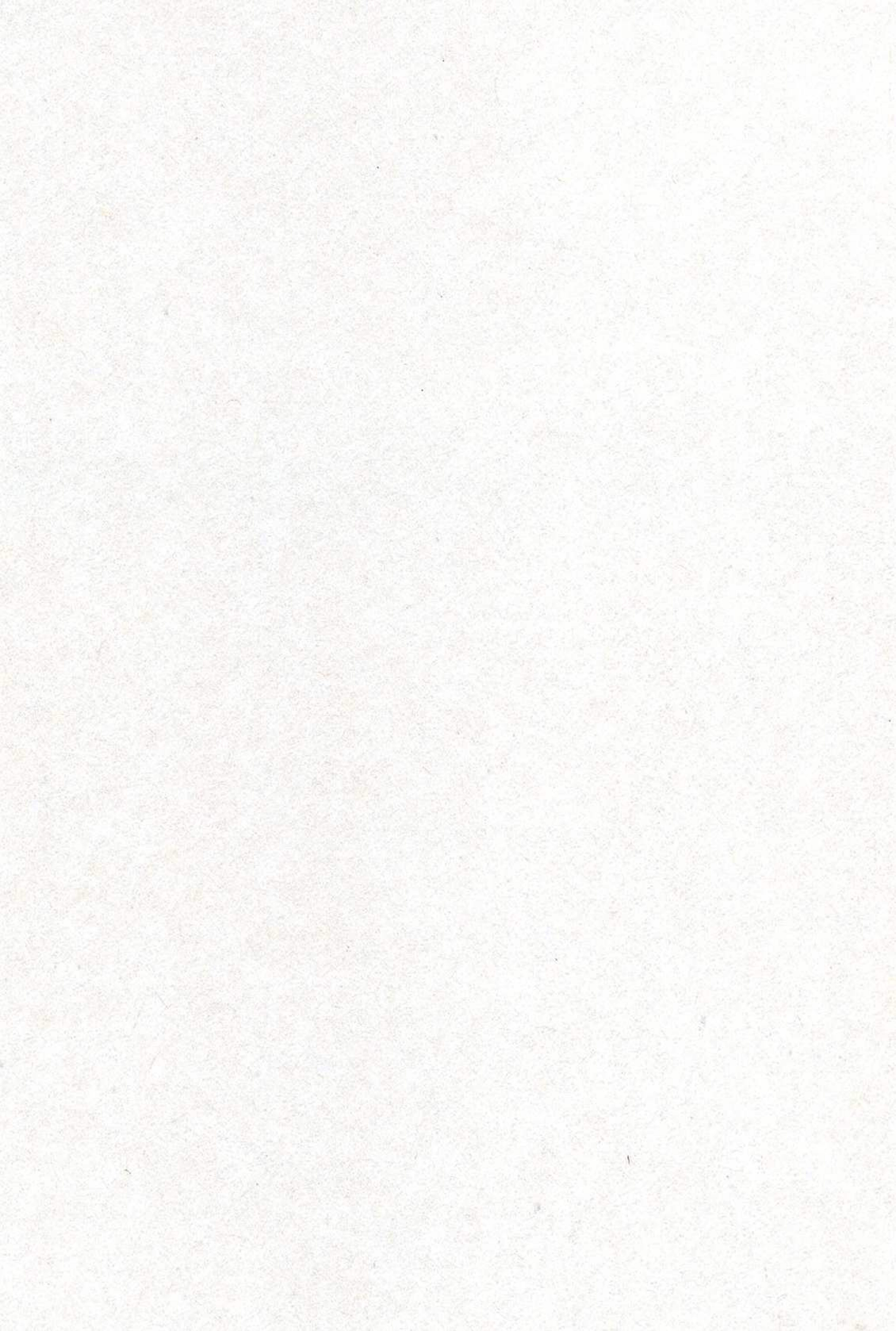
TALLER DE INFORMATICA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



TRUCOS DE PROGRAMACION Y DE DISEÑO DE GRAFICOS

EDICIONES SIGLO CULTURAL

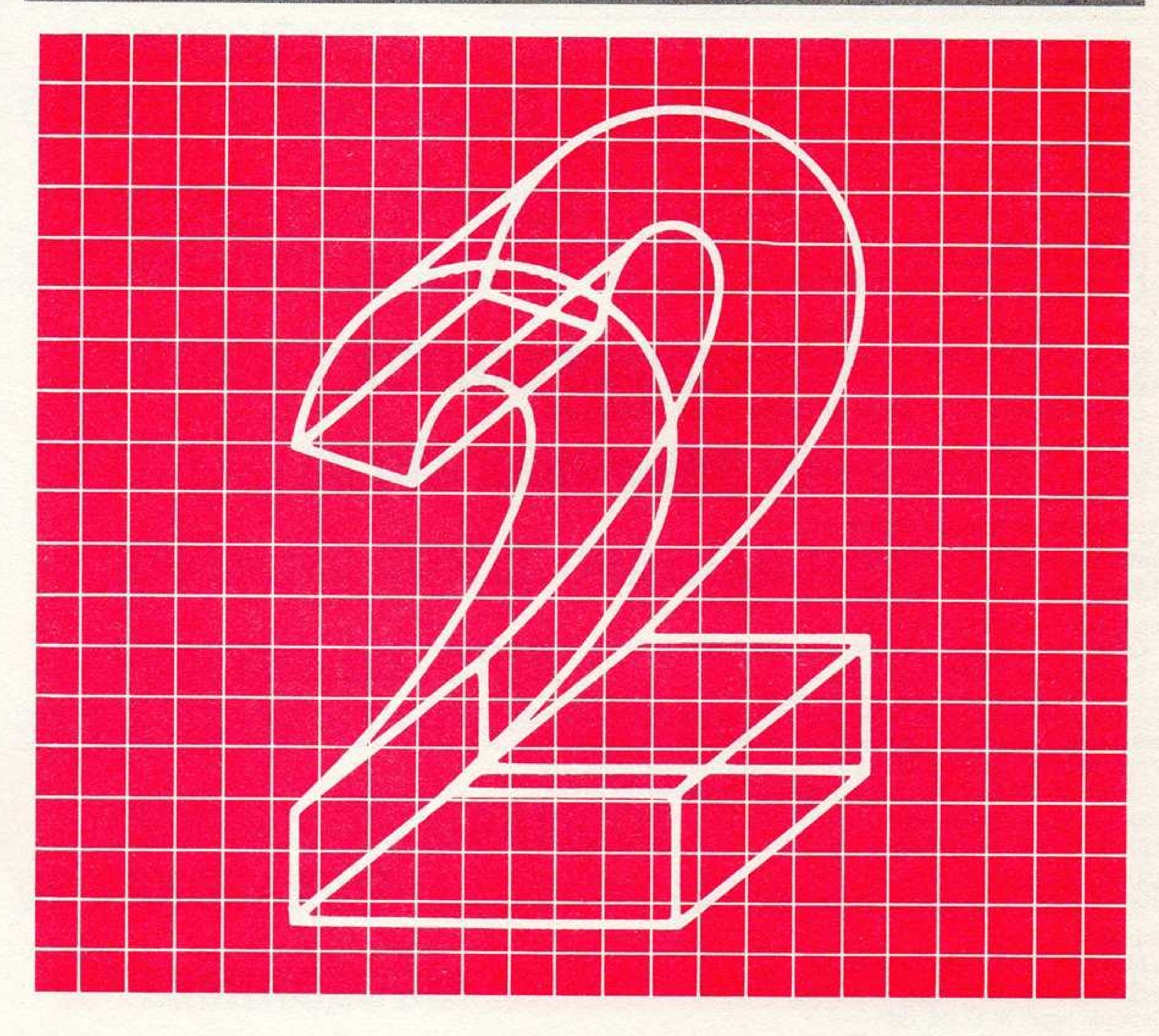


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la colecciöin:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. SIMON

Dibujos:
JOSE OCHOA

Tomo 1. «Experiencias prácticas en logo», Carlos Albert, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales, Técnico de Informática. «Aprender con el ordenador», AULA DE INFORMATICA APLICADA: Fernando Suero, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Francisco Blanca, Diplomado en Telecomunicación; María José Hernando, Diplomada en Informática; Soledad Tamariz, Diplomada en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-049-9
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

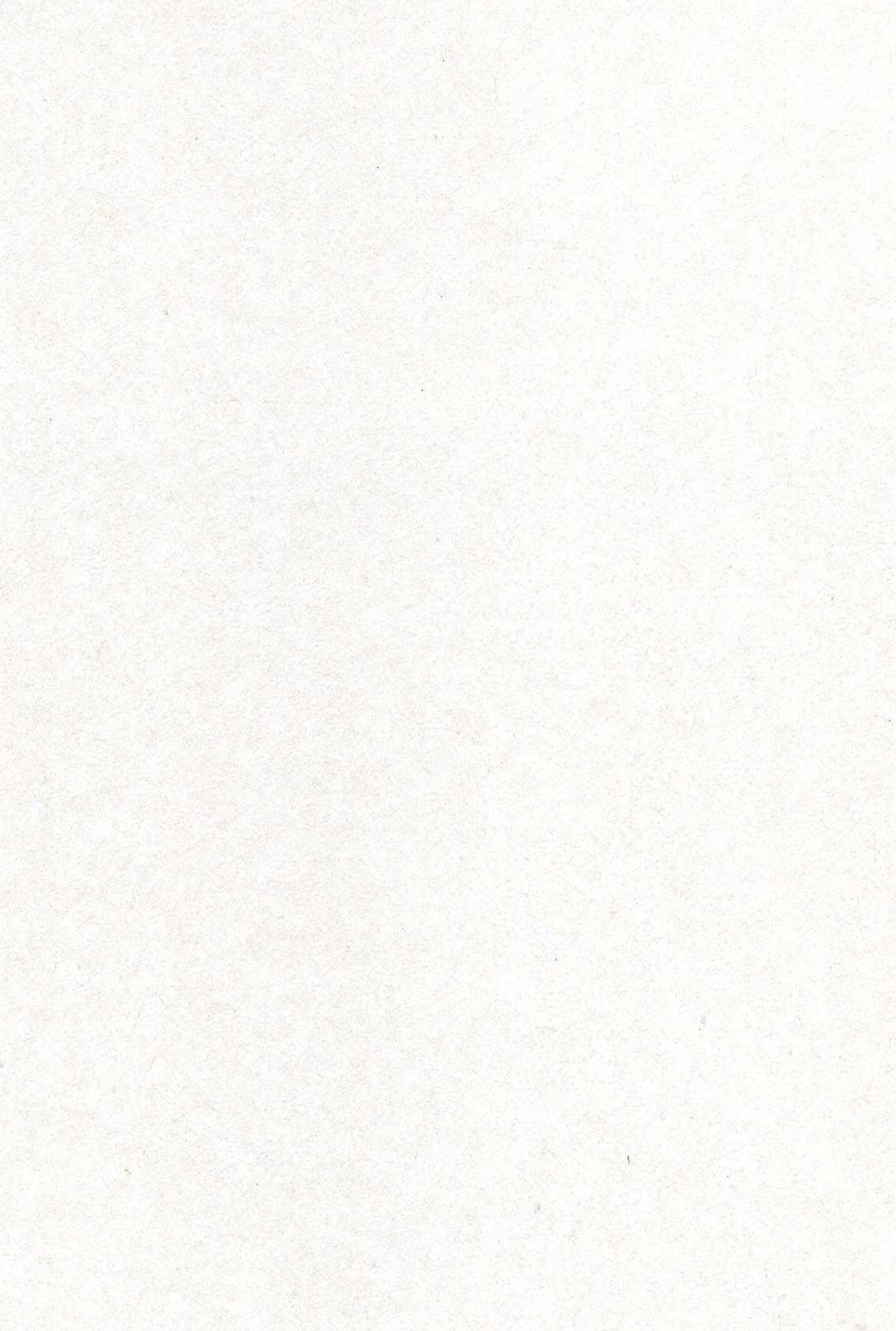
Imprime:
MATEU CROMO. PINTO (Madrid).

Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Enero, 1987.

INDICE



EXPERIENCIA Y PRACTICAS EN LOGO



A te presentamos a la Tortuga del Logo y vimos unas cuantas órdenes que servían para poder desplazarla, girarla, pintar, no pintar, tenerla oculta o visible. También vimos órdenes que borraban la pantalla y que la ponían en un modo determinado.

Con estas órdenes dibujábamos, acompañados de la Tortuga, letras y figuras geométricas.

Vamos a ver en esta ocasión un poco más despacio estas órdenes y algunas más que permiten realizar otra serie de cosas.

Recordarás que casi todas las órdenes las podíamos escribir de dos formas: abreviadas o sin abreviar.

AV = Avanza

De esta forma evitamos el tener que teclear demasiado, ahorrando espacio en memoria y tiempo.

Estas órdenes las podemos escribir en una sola línea, pero siempre tendrán que estar separadas por un espacio.

Vamos a conocer un poco mejor los diferentes tipos de pantallas que existen en el Logo.

Existen tres tipos de pantallas con las que podemos trabajar.

Nosotros ya hemos utilizado uno de estos tipos. Vamos a verlas profundizando algo más.

La orden PANTALLAMIXTA nos divide la pantalla en dos zonas diferenciadas. En una podemos escribir nuestras órdenes, mientras que en la otra sólo aparecen los gráficos que realizamos. De esta forma podemos ir tecleando las órdenes y a la vez observar cómo son realizadas.

La zona destinada a escribir se denomina zona de TEXTO y varía su tamaño dependiendo de la versión del Logo.

La orden PANTALLAGRAFICA deja toda la pantalla para los gráficos. Podemos, en este tipo de pantalla, seguir dando órdenes pero no se pueden visualizar.

Por último, la orden PANTALLATEXTO o TEXTO, deja toda la pantalla para que podamos escribir.

Ya veremos la utilidad de estos tipos de pantalla y eligiremos el más adecuado para el trabajo que realicemos.

Controlando a la Tortuga

Sabemos que la pantalla tiene un número determinado de puntos (pixels), es decir, tiene unos límites horizontales y verticales. Hemos visto cómo la Tortuga los puede sobrepasar.

Si tecleas:

```
?MUESTRATORTUGA
?BP
?VENTANA
?AV 600
```

Muchas de las órdenes en Logo tienen una contraria.

Vemos que la Tortuga se ha ido, ha atravesado los límites de la pantalla. En este caso no ha aparecido por el extremo contrario por donde ha desaparecido.

Si tecleas:

?BP

La Tortuga vuelve a la pantalla. Si ahora introduces:

?ENLAZA
?AV 600

La Tortuga ya no desaparece, sale por un extremo y vuelve por el contrario. Ocurre lo mismo que en el caso que antes decíamos.

Pero a veces, que la pantalla tenga unos límites insuperables puede sernos útil. Si tecleas:

?LIMITA
?AV 600

La Tortuga no avanza, no se mueve, el número de puntos que hemos indicado es excesivo.

Estas órdenes, VENTANA, ENLAZA y LIMITA, no existen en todas las versiones del Logo.

Vamos a conocer un poco más a fondo las órdenes que nos permitirán desplazarnos a cualquier punto de la pantalla. Para esto eran imprescindibles dos tipos de órdenes:

— Las de desplazamiento
— Las de giro

Las órdenes con que podemos desplazar la Tortuga son: AVANZA y RETROCEDE.

Con AVANZA n, movíamos a la Tortuga (n) puntos, en la dirección hacia donde estaba orientada. El valor que se le puede dar a (n) es limitado y varía según la versión con la que trabajemos. La (n) llega a admitir valores como el 25.000.

Teclea las siguientes órdenes:

?MUESTRATORTUGA
?BP
?AV 600

La Tortuga se ha desplazado los 600 puntos que le hemos ordenado. Lo único que ha ocurrido es que ha sobrepasado los límites de la pantalla y al desaparecer por el margen superior, vuelve a aparecer por el margen inferior.

¿Qué ocurre si damos la orden AV-50?

La Tortuga lo que hace es retroceder 50 puntos.

La orden RETROCEDE (RE) es la contraria a AVANZA y por ello realizará lo mismo que AVANZA, pero en el sentido opuesto.

Con la orden RE-50, lo único que hace la Tortuga es avanzar 50 puntos.

¿Y si damos unos cuantos giros?

Recordarás que utilizamos las órdenes GIRADERECHA (GD) y GIRAIZQUIERDA (GI), y que con ellas conseguíamos que la Tortuga se orientase hacia cualquier punto de la pantalla.

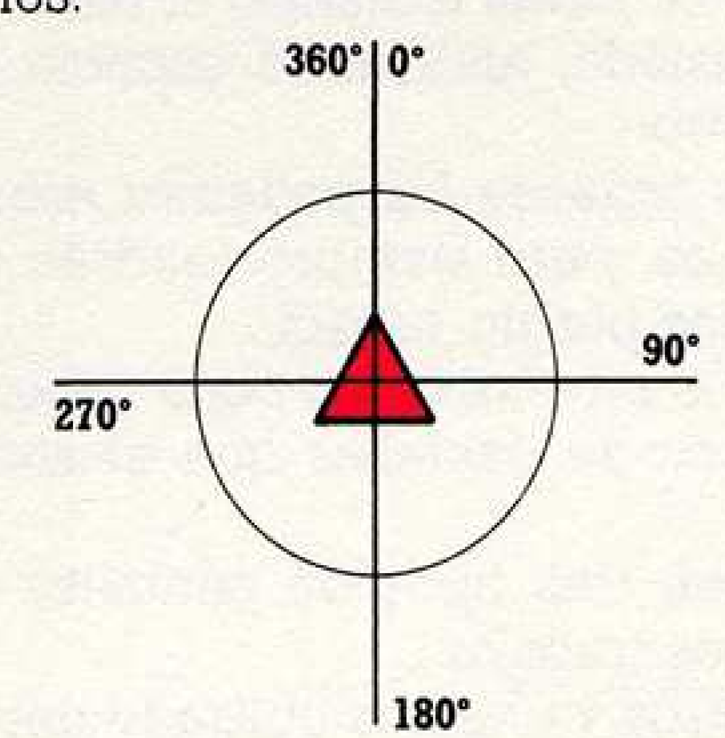
Estas dos órdenes lo que hacen es variar la orientación de la Tortuga. Lo conseguimos dando el número de grados que queremos que nos gire, bien sea a la derecha o a la izquierda. Con ello cambiamos el rumbo que tenía y podremos dirigirnos a cualquier sitio con ella.

Comprueba los siguientes giros:

? GD 90   ? GI 360
? GD 22   ? GD 8
? GI 170

Observarás que al girar 360, la Tortuga no se ha movido. La explicación es muy sencilla. Le hemos ordenado que gire una vuelta completa y, por tanto, queda orientada en la misma dirección en que estaba.

Quizá en la versión que tienes, en giros inferiores a 10 grados verás que la Tortuga tampoco se mueve. No te preocupes. No se mueve, pero el giro lo ha realizado aunque no lo veamos.



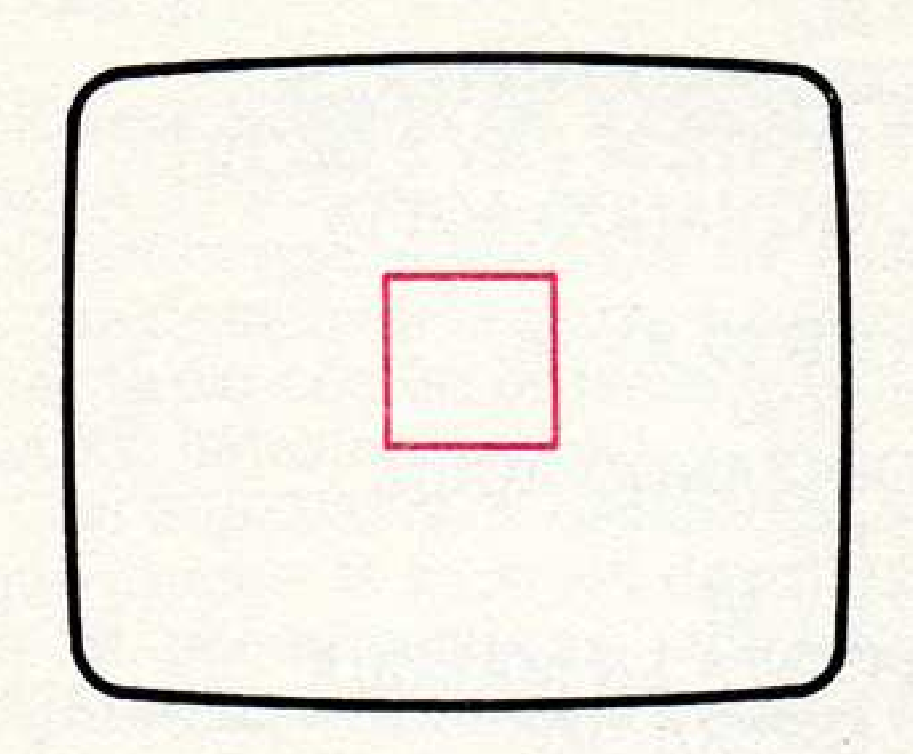
Si ves que una orden Logo no funciona en tu versión, comprueba en tu manual si existe, si no es así, busca alguna equivalente.

GD 90 le dirá a la Tortuga que varíe su rumbo 90 grados hacia la derecha.

Teclea las siguientes órdenes:

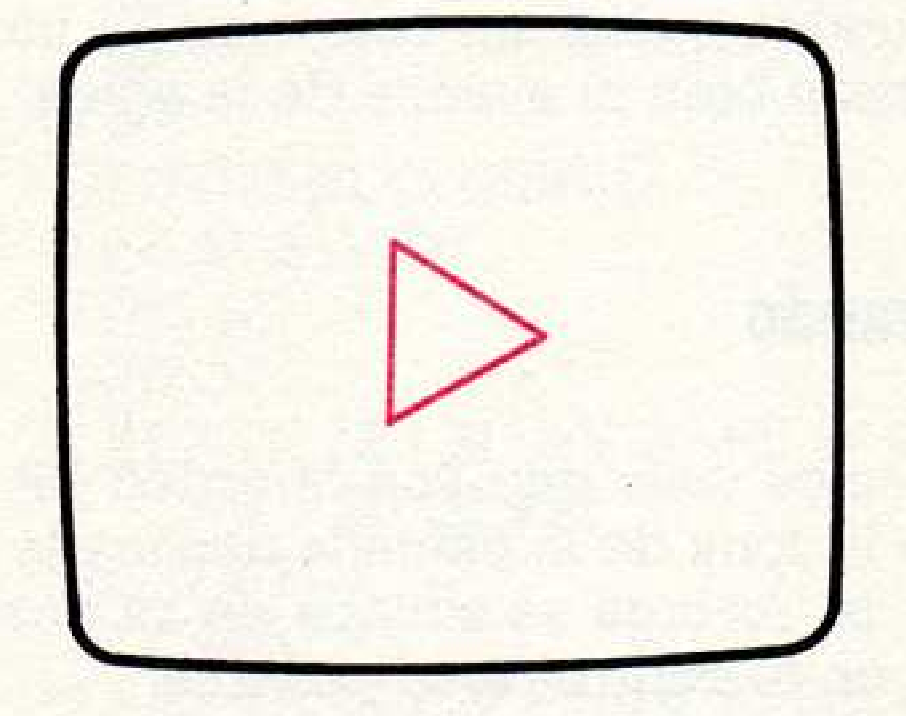
FIGURAS GEOMETRICAS
INICIALIZACION PANTALLA
Y ESTADO DE TORTUGA
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? GI 90
? AV 20
? GD 90
DIBUJANDO FIGURA
? BL
? REPITE 4 [AV 40 GD 90]

Obtendrás el siguiente dibujo:



Si vamos variando los valores de la última orden podremos obtener muchas figuras geométricas.

—REPITE 3 [AV 40 GD 120]

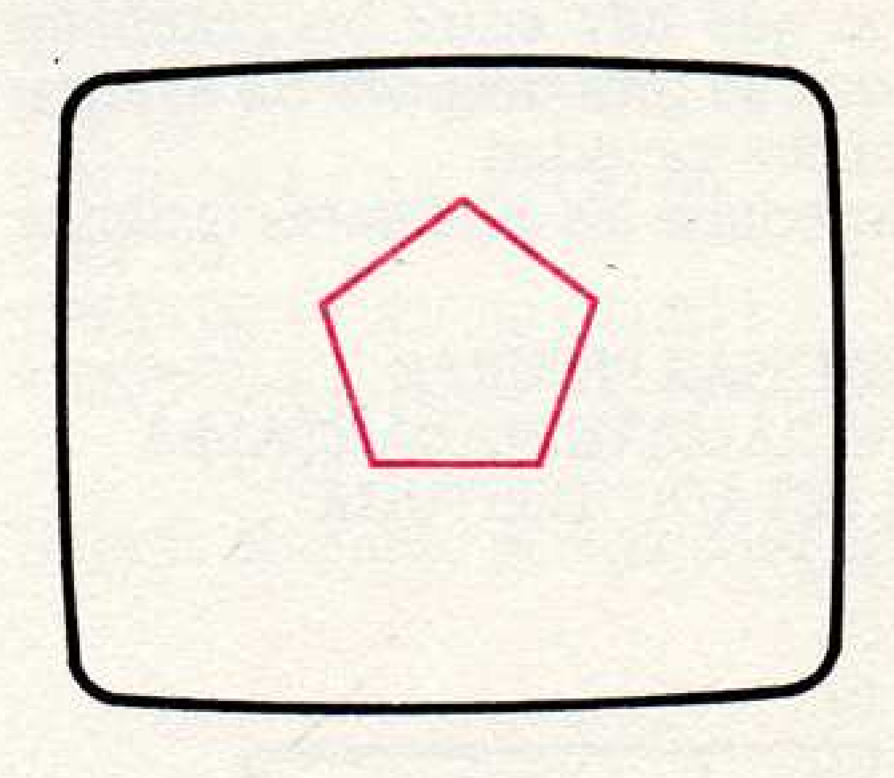


Obtendremos un triángulo equilátero.

La cantidad que estamos avanzando es el tamaño del lado de la figura. Puedes variar el tamaño de todas las figuras, cambiando el valor de la orden AV 40. Fíjate bien que en todos los casos el número que repetimos por el número que giramos, nos da 360°. Si estamos haciendo figuras geométricas cerradas, lo único que hacemos es completar un giro de 360° dividido por el número de lados que tiene la figura. La Tortuga da un giro de 360°, pero lo reparte entre el número de lados.

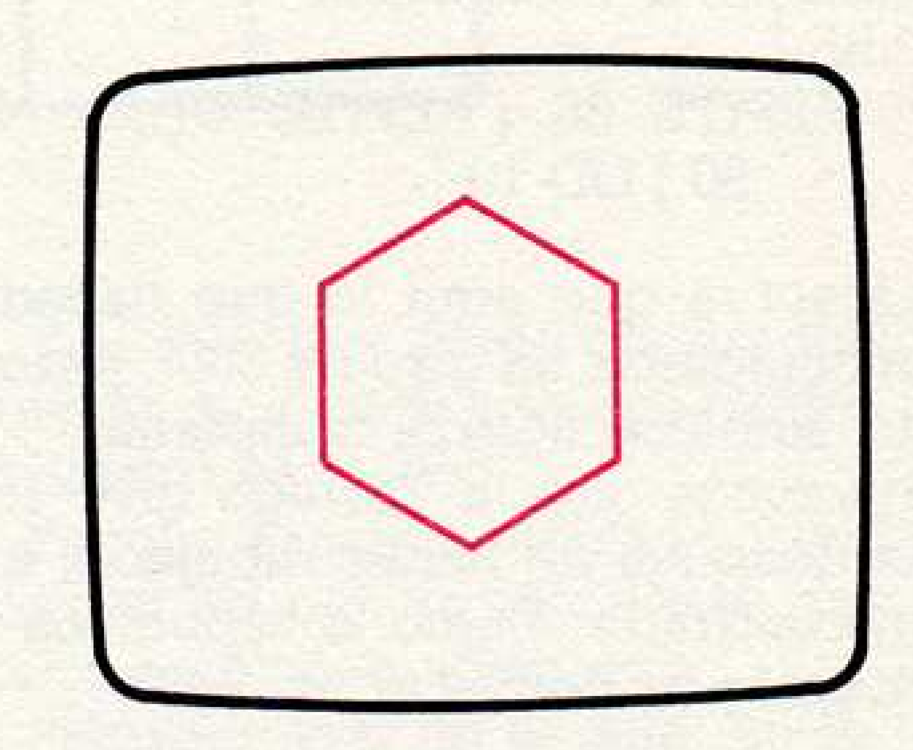
Aquí tienes una fácil forma de hacer un montón de figuras geométricas regulares.

—REPITE 5 [AV 40 GD 72]



Obtendremos un pentágono.

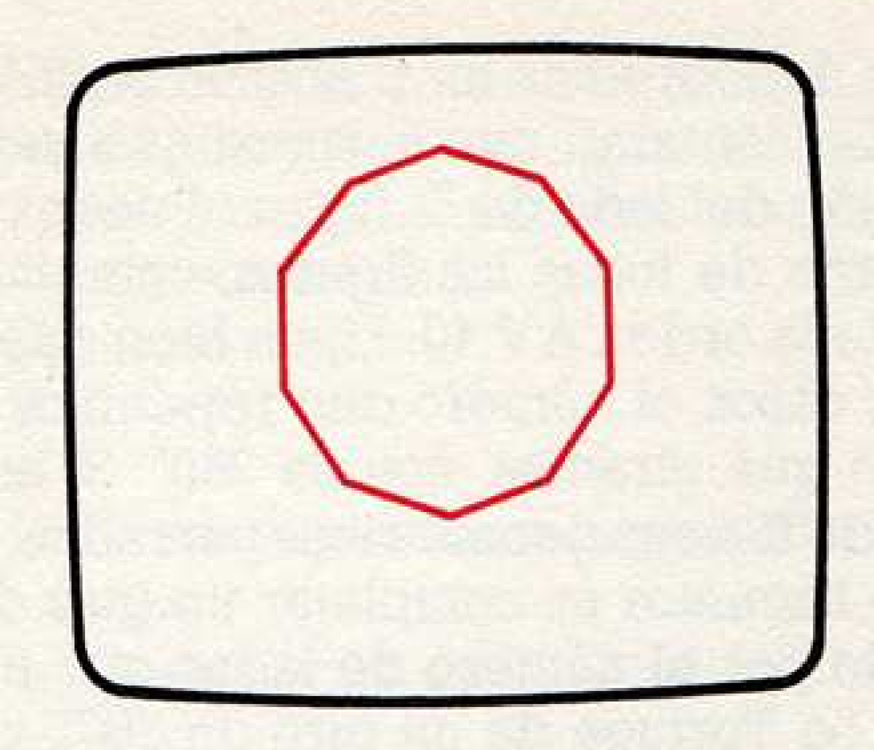
—REPITE 6 [AV 40 GD 60]



Obtendremos un hexágono.

—REPITE 10 [AV 40 GD 36]

Si vas a hacer un dibujo complicado es mejor que te hagas primero un pequeño esquema sobre el papel.



Obtendremos un decágono.

Observa que lo único que variamos en la orden es el número de veces que vamos a repetir lo que hay entre los corchetes y los grados que se gira después de avanzar siempre la misma cantidad.

Introduce las siguientes órdenes:

```
MUCHAS FIGURAS
INICIALIZANDO PANTALLA
Y ESTADO TORTUGA
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? GI 90
? AV 20
? GD 90
DIBUJANDO FIGURAS
? BL
? REPITE 36 [REPITE 4 [ AV 40 GD
      90 ] GD 10 ]
```

Observa que aquí lo que hacemos es también completar un giro de 360°, pero ya no con lado, sino con figuras completas.

Hacemos que la Tortuga gire 10° a la derecha después de haber dibujado, en este caso, un cuadrado. Como lo repetimos 36 veces, volvemos a completar los 360°

$10 \times 36 = 360$

Obtendrás bonitos dibujos como éste si cambias en este programa las órdenes que pinta el cuadrado (REPITE 4 [AV 40 GD 90]), por cualquiera de las órdenes que has introducido antes para otras figuras.

Puedes también cambiar el tamaño y el número de figuras que quieres que compongan tu dibujo.

Introduce ahora las siguientes órdenes y conseguirás un cronómetro que más adelante podrás ir perfeccionando.

```
CRONOMETRO
INICIALIZA PANTALLA
Y ESTADO TORTUGA
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? GI 90 AV 50
? GD 90
? BL
DIBUJANDO EXTERIOR
? REPITE 36 [ AV 8 GD 10 ]
CENTRANDO AGUJA
? SL
? GD 90 AV 46
? GI 90
DIBUJANDO AGUJA
? BL
? AV 44
PUESTA EN MARCHA
? ESPERA 100
? GOMA
? RE 44 SL AV 4
? REPITE 180 [ BL AV 40 GOMA RE 40
      AV 40 RE 40 GD 2 ]
PARADA
? BL
? AV 40
```

Puedes variar en este caso el número de vueltas o bien el avance de la aguja.

Borrando

Hemos visto que con la orden BP, tras borrarse la zona de la pantalla destinada a los gráficos, la Tortuga se situaba en su posición original, en el centro de la pantalla.

En Logo de los MSX, vienen definidas un montón de tortugas. Si quieres verlas da la orden PONFORMA (n).

Con la orden LIMPIA también lo borramos todo, pero en este caso la posición de la Tortuga no varía.

Pero qué pasa si queremos borrar sólo algo que nos ha salido mal o que no nos gusta. Sería absurdo tener que borrar todo y volver a comenzar. Para estos casos utilizaremos la orden:

GOMA

Esta orden activa la goma que tiene la Tortuga. Con la goma activada, ya podremos borrar todo lo que queramos. Simplemente tendremos que pasar por encima de lo que nos disponemos a borrar.

Comprúebalo tecleando:

```
? AV 60
GOMA
? RE 30
```

Hemos borrado un tramo de la recta que hemos dibujado.

Cuando queramos dejar de borrar, tenemos que desactivar la goma, soltarla. Esto lo podemos hacer con las órdenes SUBELAPIZ o BAJALAPIZ, indistintamente.

Con SUBELAPIZ, si ordenamos a la Tortuga que avance, lo hace sin dejar rastro, sin dibujar. Si, por el contrario, utilizamos la orden BAJALAPIZ, la Tortuga al avanzar va dibujando.

Introduce las siguientes órdenes:

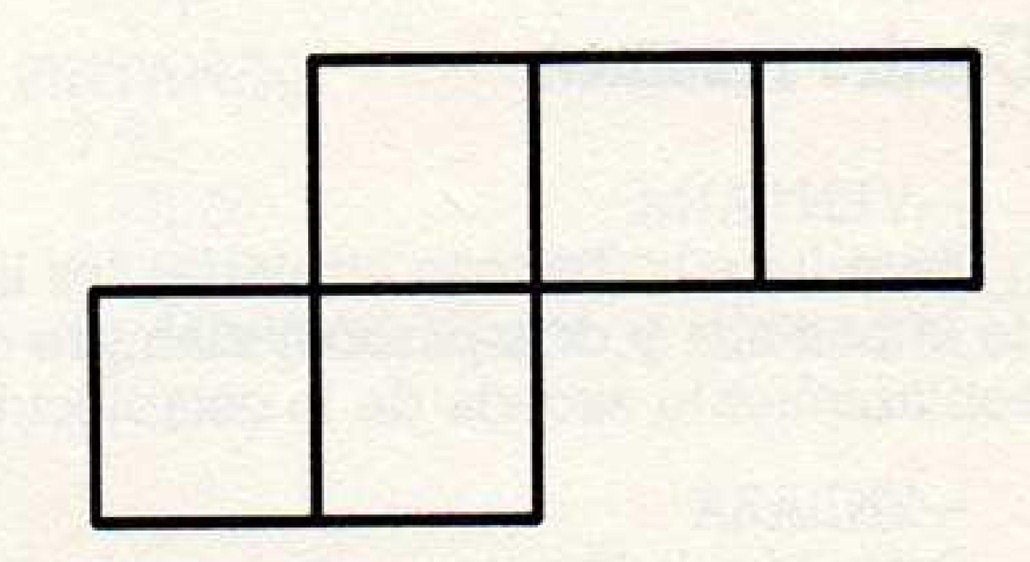
```
VISTO Y NO VISTO
INICIALIZANDO PANTALLA
Y ESTADO TORTUGA
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? GI 90 AV 100
? GD 90
VISTO
? REPITE 12 [ BL AV 20 RE 20 GD 90 SL
        AV 20 GI 90 ]
NO VISTO
? GOMA
? REPITE 12 [ GI 90 AV 20 GD 90 AV 20
        RE 20 ]
```

Con las órdenes que hemos visto hasta ahora estoy seguro que serás capaz de resolver el siguiente juego:

Teclea las órdenes:

```
JUEGO CUADRADOS
INICIALIZA PANTALLA
Y ESTADO TORTUGA
? PM
? SL
? BT
? OT
CENTRANDO DIBUJO
? RE 50 GI 90
? AV 100 GD 90
DIBUJANDO 3 CUADRADOS
? BL
? REPITE 4 [ AV 50 GD 90 ]
? GD 90 AV 50
? GI 90
? REPITE 4 [ AV 50 GD 90 ]
? AV 50
? REPITE 4 [ AV 50 GD 90 ]
? REPITE 2 [ GD 90 AV 50 GI 90 REPITE
        4 [ AV 50 GD 90 ] ]
```

Habrás obtenido el siguiente dibujo:



Ves que hay cinco cuadrados. Intenta transformar moviendo sólo dos palitos, los cinco cuadrados en cuatro.

Es muy sencillo, utiliza la orden GOMA para borrar los palitos que tienes que borrar, y píntalos en el sitio adecuado.

Piénsalo, y si no te sale, no te preocupes, aquí tienes una posible solución:

```
SOLUCION
POSICION PARA BORRAR
UN PALITO
? SL
```

El Logo de los PC compatibles permite las órdenes: VENTANA, LIMITA y ENLAZA.

? CENTRO
? AV 50 GD 90
BORRANDO UN PALITO
? GOMA
? AV 49
POSICION PARA PINTARLO EN OTRO SITIO
? CENTRO
? BL
? RE 50 GD 90
PINTANDOLO EN OTRO SITIO
? AV 50
POSICION PARA BORRAR OTRO PALITO Y BORRARLO
? RE 50
? GOMA
? RE 49
POSICION PARA PINTARLO EN OTRO SITIO
? SL
? AV 99
? GI 90
PINTANDOLO EN OTRO SITIO
? BL
? AV 50

Cuadro resumen

—VENTANA
Permite a la Tortuga atravesar los límites de la pantalla y desaparecer siempre que el desplazamiento exceda de lo establecido.

—ENLAZA
Permite a la Tortuga volver a aparecer por el extremo opuesto por el que salió.

—LIMITA
No permite a la Tortuga sobrepasar los límites de la pantalla cuando el desplazamiento sea excesivo.

—LIMPIA
Borra la zona de la pantalla destinada a los gráficos dejando a la Tortuga en la posición en que se encontraba.

—GOMA
Activa la Goma de la Tortuga para poder borrar aquellas zonas por donde la Tortuga pase, siempre que haya algo dibujado debajo.

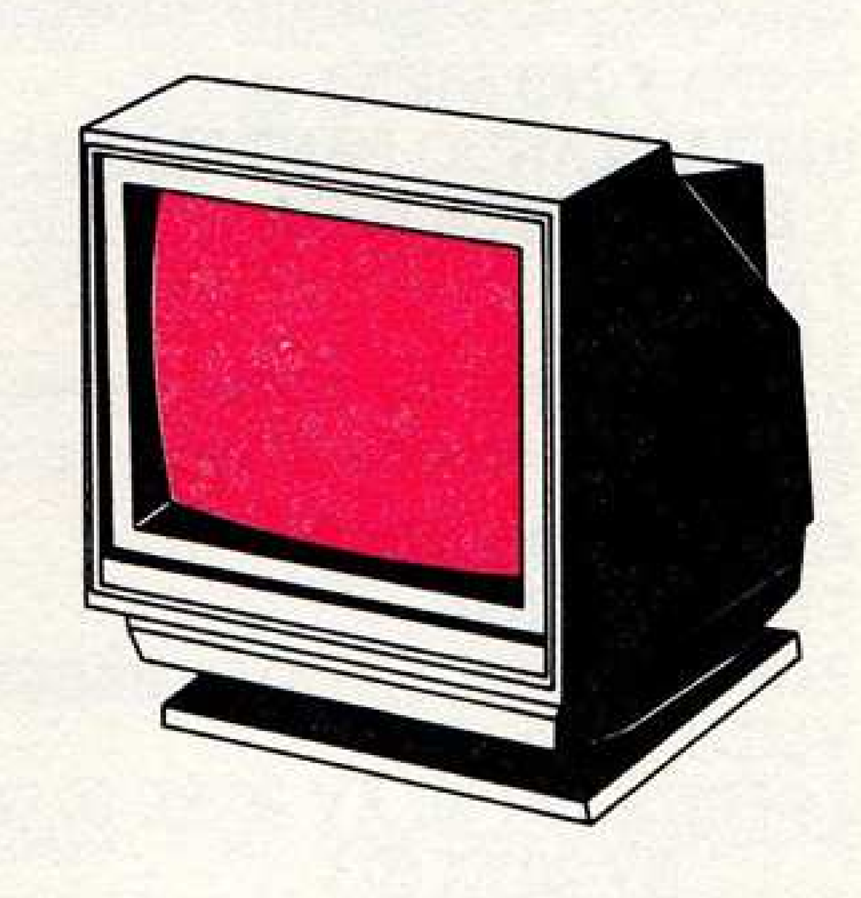
—PANTALLAGRAFICA (PG)
Sitúa la pantalla en modo gráfico, dejando toda la superficie para textos y gráficos entremezclados.

—PANTALLATEXTO
Sitúa la pantalla en modo texto, dejando toda la superficie para escribir.

—CENTRO
Sitúa a la Tortuga en el centro de la pantalla, desplazándola desde el punto en donde se encuentre.

Ejercicios

1) *Dibuja con la tortuga, un rombo, un trapecio y un rectángulo.*
2) *Dibuja una circunferencia de cada una de las figuras anteriores, pero conteniendo sólo 10 figuras cada una.*
3) *¿Son correctas estas órdenes?*
   - PANTALLAM
   - CENTRO 25
   - LIMPIA
   - GOMAAV 25
   - GD [45]
   - VENTANA
4) *Haz el siguiente dibujo.*



5) *Dibuja una flecha que se vaya desplazando de un extremo a otro de la pantalla.*

Con el LOGO puedo borrar utilizando la goma.

Solución a los ejercicios

1:

a) ROMBO

```
INICIALIZANDO PANTALLA
Y ESTADO DE LA TORTUGA
  ? PM
  ? SL
  ? BP
  ? OT
CENTRANDO DIBUJO
  ? GD 180 AV 30
  ? GD 160
PINTANDO
  ? BL
  ? REPITE 2 [ REPITE 2 [ AU 30 GD
  40 ] GD 100
```

b) TRAPECIO

```
INICIALIZANDO PANTALLA
Y ESTADO TORTUGA
  ? PM
  ? SL
  ? BP
  ? OT
CENTRANDO DIBUJO
  ? GD 90 AU 30
  ? GI 180
PINTANDO
  ? BL
  ? AU 60 GD 135
  ? AU 30 GD 45
  ? AU 18 GD 45
  ? AU 30
```

c) RECTANGULO

```
INICIALIZANDO PANTALLA
Y ESTADO TORTUGA
  ? PM
  ? SL
  ? BP
  ? OT
CENTRANDO DIBUJO
  ? GP 90 AV 30
  ? GI 180
PINTANDO
  ? BL
  ? AV 60 GD 90
  ? AV 30 GD 90
  ? AV 60 GD 90
  ? AV 30 GD 90
```

2:

```
ROMBOS
INICIALIZANDO PANTALLA
Y ESTADO DE LA TORTUGA
  ? PM
  ? BP
  ? OT
PINTADO
  ? BP
  ? REPITE 10 [ REPITE 2 [ REPITE
  2 [AU 30 GD 40 ] GD 100 ] GD 36
TRAPECIO
INICIALIZACION PANTALLA
Y ESTADO TORTUGA
  ? PM
  ? BP
  ? OT
PINTANDO
  ? BL GI 20
  ? REPITE 10 [ AV 60 GD 135 AV 30 GD
  45 AV 18 GD 45 AV 30 GI 225 GD 36 ]
TRIANGULO
INICIALIZANDO PANTALLA
Y ESTADO TORTUGA
  ? PM
  ? BP
  ? OT
PINTANDO
  ? BL
  ? GI 90
  ? REPITE 10 [ QV 60 GD 90 AV 30 GD
  90 AV 60 GD 90 AV 30 GD 90 GD 36 ]
```

3:

PANTALLAM: INCORRECTA. Esta orden no existe. La correcta es PANTALLAMIXTA o PM.
CENTRO 35: INCORRECTA. Esta orden no admite argumento.
LIMPIA: CORRECTA.
GOMAAV 25: INCORRECTA. Tiene que haber un espacio entre Goma y AV 25.
GD [45]: INCORRECTA. La forma exacta sería GD 45. El argumento no puede ir entre corchetes.
VENTANA: CORRECTA.

Puedo hacer una programación más estructurada que con otros lenguajes.

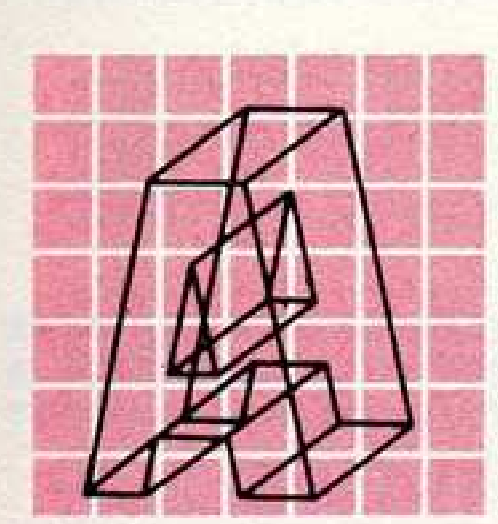
4:

```
TELEVISOR
    INICIALIZANDO PANTALLA
    Y ESTADO TORTUGA
      ? PM
      ? SL
      ? BP
      ? OT
    CENTRANDO DIBUJO
      ? GI 90 RE 25
    DIBUJANDO
    CAJA TV
      ? BL
      ? AV 50 GD 90
      ? AV 60 GD 90
      ? AV 50 GD 90
      ? AV 60 GI 135
      ? AV 25 GI 45
      ? AV 60 GD 135
      ? AV 25 GD 45
      ? AV 50 GD 135
      ? AV 25 GD 45
      ? AV 50
    PANTALLA
      ? SL
      ? CENTRO
      ? AV 15 GI 90 RE 20
      ? BL
      ? REPITE 4 [ AV 40 GD 90 ]
    MANDOS
      ? SL
      ? AV 40 GD 90 RE 10
      ? BL
      ? REPITE 3 [ REPITE 4 [ AV 5 GD
      90 ] SL GD 90 AV 8 GI 90 BL ]
      ? AV 5 GD 90
      ? AV 15 GD 90
      ? AV 5 GD 90
      ? AV 15 GD 90
```

5:

```
FLECHA
    INICIALIZANDO PANTALLA
    Y ESTADO TORTUGA
      ? PM
      ? SL
      ? BP
      ? OT
    CENTRANDO FLECHA
      ? GD 90 RE 127
    PINTANDO Y MOVIENDO
      ? BL
      ? REPITE 45 [ LIMPIA AV 25 GD 45 RE
      5 AV 5 GI 90 RE 5 AV 5 GD 45 RE 20 ]
```

La forma en que doy las órdenes es clara y se semejan a mi forma normal de decirlo.



NTERIORMENTE vimos algo sobre la definición de caracteres. Explicamos cómo se almacenaban los caracteres en la memoria del ordenador y empezamos a definir nuestros propios caracteres.

En este capítulo veremos cómo terminar de definirlos, cómo almacenarlos y cómo utilizarlos.

Ya dijimos que cogieses un lápiz y un papel cuadriculado. Con ello podremos dibujar nuestro primer carácter. Uno de los caracteres que podemos definir es la pelota que aparece en la figura 1.

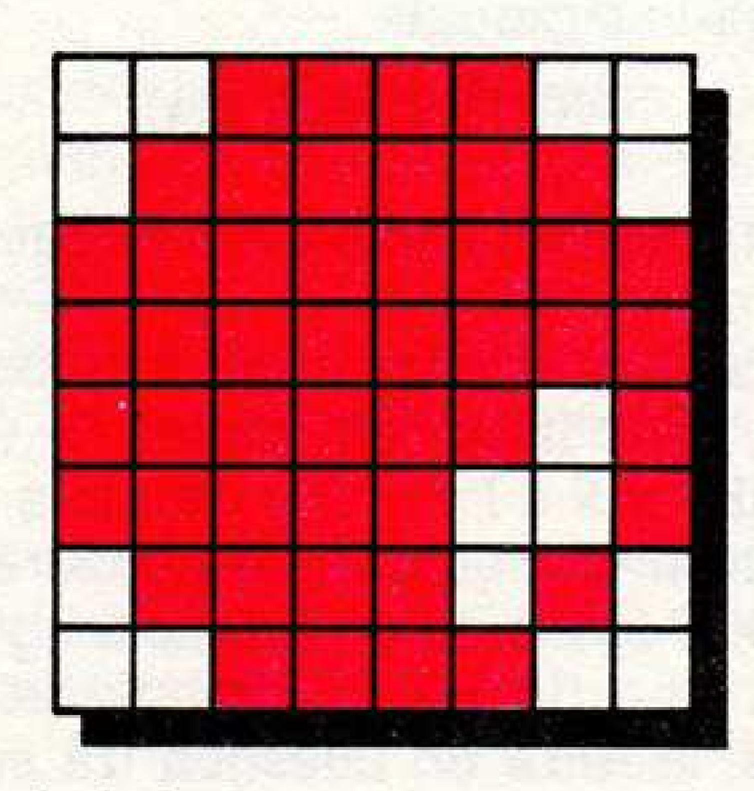


Fig. 1.—Retícula de 8 × 8 puntos con la definición de una pelota.

Una vez dibujada la pelota se transforman los cuadrados blancos en ceros y los negros en unos, obteniéndose algo parecido a lo que aparece en la figura 2.

Como cada línea tiene ocho números (entre ceros y unos), podemos pasar este número, que está en binario, a base 10. Cuando

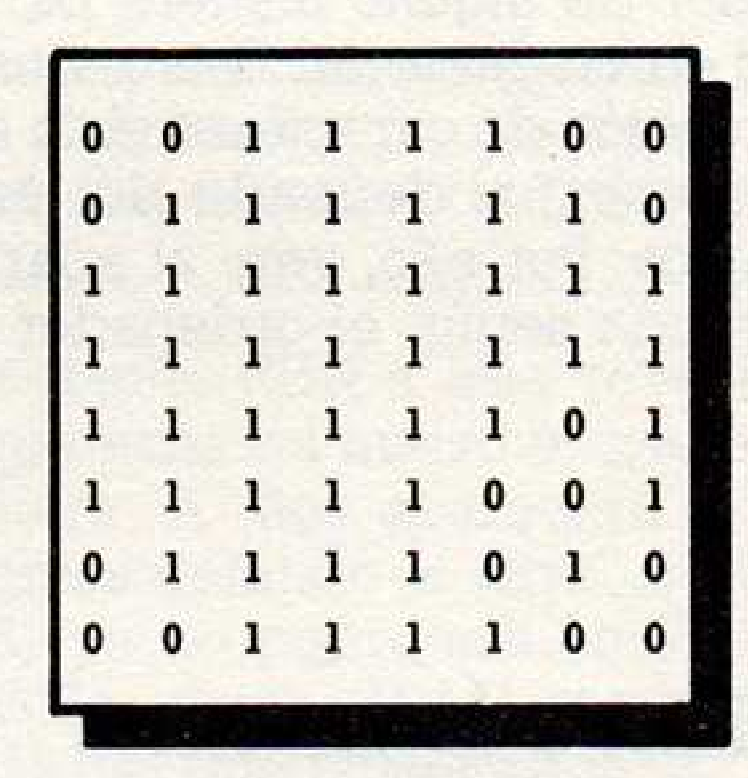


Fig. 2.—Así queda la malla de 8 × 8 puntos al sustituir los cuadrados blancos por ceros y los negros por unos.

lo hayamos hecho tendremos ocho números (porque teníamos ocho filas). Para pasar estos números en binario a base 10 puedes utilizar el programa 1 que se da en la sección de TRUCOS DE PROGRAMACION, uniéndolo al programa 1 de esta sección.

```
 10 REM ****************************
 20 REM * PASO DE BASE 2 A BASE 10 *
 30 REM ****************************
 40 REM
 50 DIM N(8):LET B1=2
 60 FOR I=1 TO 8
 70    PRINT "DAME LA FILA ";I;
 80    INPUT N(I)
 90    GOSUB 8900
100    LET N(I)=NU
110 NEXT I
120 REM
130 REM *** IMPRESION DE LOS NUMEROS ***
140 REM
150 PRINT "LOS NUMEROS EN BASE 10 SON :"
160 PRINT
170 FOR I=1 TO 8
180    PRINT N(I);" ";
190 NEXT I
200 END
```

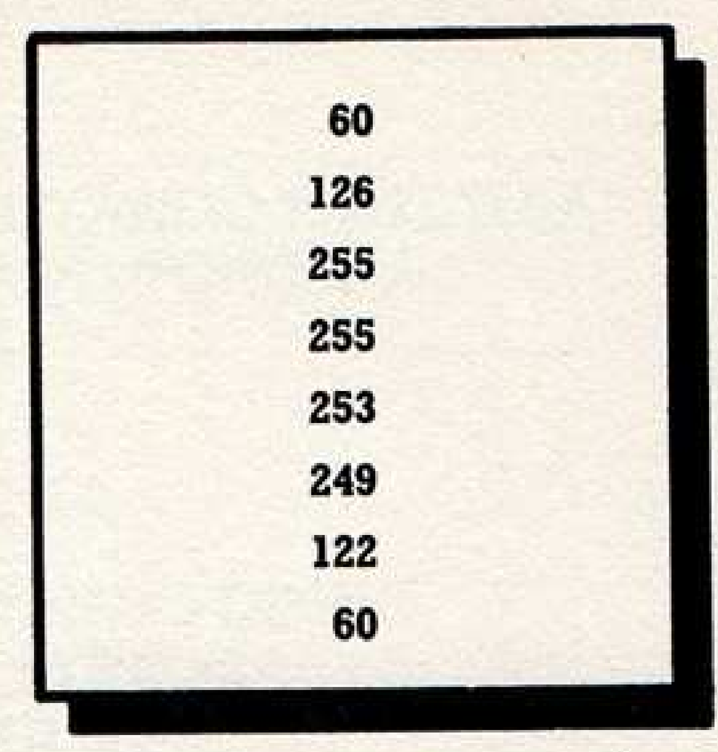


Fig. 3.—Estos son los números una vez pasados a base 10.

Una vez que tenemos los números preparados (ver figura 3), hay que meterlos en el ordenador de alguna manera para que se conviertan en un carácter. Esta parte del proceso de creación de caracteres es la más compleja (no mucho) y depende del ordenador que tengamos. Por ello, vete al apartado que te corresponda según el ordenador que tengas.

Definición de caracteres en el Spectrum

En el Spectrum tenemos 21 posibles gráficos definibles por el usuario (GDU). Estos son los que están comprendidos entre la A y la U. Para averiguar las direcciones de memoria donde empieza la definición de cada letra utilizaremos la función USR. Si escribimos en nuestro ordenador:

PRINT USR "A"

nos aparecerá en pantalla el número 64349. Este número nos dice la dirección del primer BYTE (fila) del carácter 'A'. No hay que confundir este carácter con la 'A', cuyo código ASCII es 65. Esta 'A' tiene el código ASCII 144 y es la que podremos definir.

Continuando lo que decíamos antes, el número 64349 nos da el primer BYTE de la 'A', el 64350 el segundo, etc. Y así hasta el 64356, que almacena el último BYTE de la 'A'.

Para introducir cada número de la definición de la pelota en el lugar que ocupa la definición de la 'A', utilizaremos, como vimos en el fascículo anterior, la sentencia POKE.

En el SPECTRUM podemos utilizar el siguiente formato:

POKE DM,BIN No.

donde DM es la dirección de memoria donde queremos meter el número y BIN No es el número que queremos introducir, pero escrito en base 2. Un ejemplo sería:

POKE 64349,BIN 00111100

que correspondería a poner en la primera fila de nuestro carácter definible 'A' la primera fila de la pelota que queremos definir.

Sabiendo esto, ya podemos introducir en nuestro ordenador la definición del primer carácter diseñado por nosotros. Para ello utilizaremos el programa 2.

```
10 REM ********************************************
20 REM * DEFINICION DE UNA PELOTA EN EL SPECTRUM *
30 REM ********************************************
40 REM
50 POKE USR "A"+0,BIN 00111100
60 POKE USR "A"+1,BIN 01111110
70 POKE USR "A"+2,BIN 11111111
80 POKE USR "A"+3,BIN 11111111
90 POKE USR "A"+4,BIN 11111101
100 POKE USR "A"+5,BIN 11111001
110 POKE USR "A"+6,BIN 01111010
120 POKE USR "A"+7,BIN 00111100
```

Para ver el carácter que hemos introducido teclea lo siguiente:

PRINT CHR$(144)

y nos aparecerá en pantalla una pequeña pelotita.

Otra forma de imprimir este carácter es poniendo el cursor en modo G (pulsando CAPS-SHIFT y 9 a la vez) y dándole a la tecla 'A'. Este será el método que utilizaremos normalmente. Cuando en un listado tengamos que introducir uno de los 21 caracteres gráficos definibles, la letra en cuestión irá subrayada de la siguiente manera:

PRINT "<u>A</u>"

El programa que hemos utilizado para crear la pelotita es un poco rupestre y comilón de memoria. Otro programa podría ser:

Este programa es más corto (por lo que ahorra más memoria) y más bonito como programa (está mejor estructurado).

Te aconsejo que, antes de continuar, te construyas tus propios caracteres gráficos para que tengas más práctica. Como ejemplo te propongo los caracteres de la figura 4 aunque siempre es conveniente que tú mismo los inventes y dibujes.



Fig. 4.—Estos son algunos de los caracteres que puedes realizar para ir cogiendo práctica.

Más adelante veremos cómo redefinir el juego entero de caracteres del SPECTRUM mediante un truco. Veremos cómo hacen los programas comerciales para tener muchos caracteres definidos y para escribir con letras del Oeste, del espacio o incluso en fenicio.

Definición de caracteres en el Amstrad

Al ser el AMSTRAD un ordenador más potente que el SPECTRUM, la forma de definir caracteres es más sencilla. En el AMSTRAD se puede definir un carácter en una sola línea mediante la sentencia SYMBOL, cuyo formato es:

SYMBOL C (lista de ocho números)

donde C es el código ASCII del carácter que queremos definir y lista de ocho números son los ocho números que corresponden a las ocho filas del nuevo carácter a definir.

Según esto, como podemos apreciar, en el AMSTRAD todos los caracteres son definibles (todos menos los 32 primeros que son de control).

En el AMSTRAD, para definir la pelotita de la figura 1 sólo habría que escribir el siguiente programa:

```
10 REM ******************************************
20 REM * DEFINICION DE UNA PELOTA EN EL SPECTRUM *
30 REM ******************************************
40 REM
50 FOR I=USR"A" TO USR"A"+7
60    READ N
70    POKE I,N
80 NEXT I
90 REM
100 REM *** LINEA DE DATAS ***
110 REM
120 DATA 60,126,255,255,253,249,122,60
```

Haciendo RUN a este programa, cuando nos aparezca de nuevo el cursor, si pulsamos la letra 'A' nos aparecerá una pelotita. Como hemos definido la letra 'A' (con código ASCII 65) ya no podremos escribir palabras como DATA, como READ o como LOCATE, pues cada vez que pulsemos la 'A', nos aparecerá una pelota. La forma de solucionar este problema es utilizar, a la hora de definir nuestros caracteres gráficos, caracteres que no utilizemos normalmente. Como tenemos 255, y los más utilizados son las letras, los números y los signos de puntuación, lo normal será utilizar caracteres cuyo código ASCII sea mayor que el 126.

Una vez que hemos definido un carácter no podremos volver a tener en su lugar el que era anteriormente. Si quieres que la tecla 'A' sea otra vez la letra 'A', desconecta y vuelve a conectar tu ordenador.

```
10 REM ******************************************
20 REM * DEFINICION DE UNA PELOTA EN EL AMSTRAD *
30 REM ******************************************
40 REM
50 SYMBOL AFTER 65
60 SYMBOL 65,60,126,255,255,253,249,122,60
70 END
```

En la línea 50 del programa 4 podemos ver que pone SYMBOL AFTER 65. ¿Qué significa esto?

Si no ponemos esta instrucción el AMS-

TRAD no nos dejaría definir caracteres cuyo código ASCII fuese menor que 240. Así, al ponerla, nos deja definir todos los caracteres con códigos comprendidos entre 65 y 255. Su sintaxis es:

SYMBOL AFTER n

donde n es el primer carácter que queremos que sea definible.

Esta particularidad del AMSTRAD nos permitirá, según vayamos aprendiendo, definirnos un nuevo alfabeto con letras del Oeste, espaciales o incluso fenicias o egipcias.

Antes de continuar te recomiendo que pruebes a definir más caracteres gráficos. Como ayuda puedes intentar definir los que se dan en la figura 4, pero te recomiendo que te inventes otros y pruebes con ellos.

Definición de caracteres en el IBM

En el IBM no se pueden definir caracteres gráficos distintos de los que tiene. De todas maneras, hay una serie de trucos que nos permitirán definir figuras y moverlas por la pantalla, pero su dificultad hace necesario posponerlo para cuando dominemos en más profundidad lo que estamos viendo y la alta resolución.

Definición de caracteres en el Commodore

La definición de nuevos caracteres en el COMMODORE es bastante compleja debido a que los caracteres estándar se encuentran almacenados en memoria ROM (memoria de sólo lectura) y, por tanto, no podemos variarlos. Aun así, según vaya avanzando la obra, veremos cómo podemos engañar al ordenador para tener nuestro propio juego de caracteres.

Aunque no podamos definir caracteres, sí podemos definir lo que se denomina SPRITES. Un SPRITE ('duendecillo' en inglés) es una figura diseñada por nosotros y que se podrá mover por la pantalla de una forma más bonita y cómoda que si utilizásemos caracteres.

Los SPRITES del COMMODORE se definen mediante una malla de 24 por 21 puntos (no de 8 por 8 como los caracteres) como aparece en el dibujo de la figura 5.

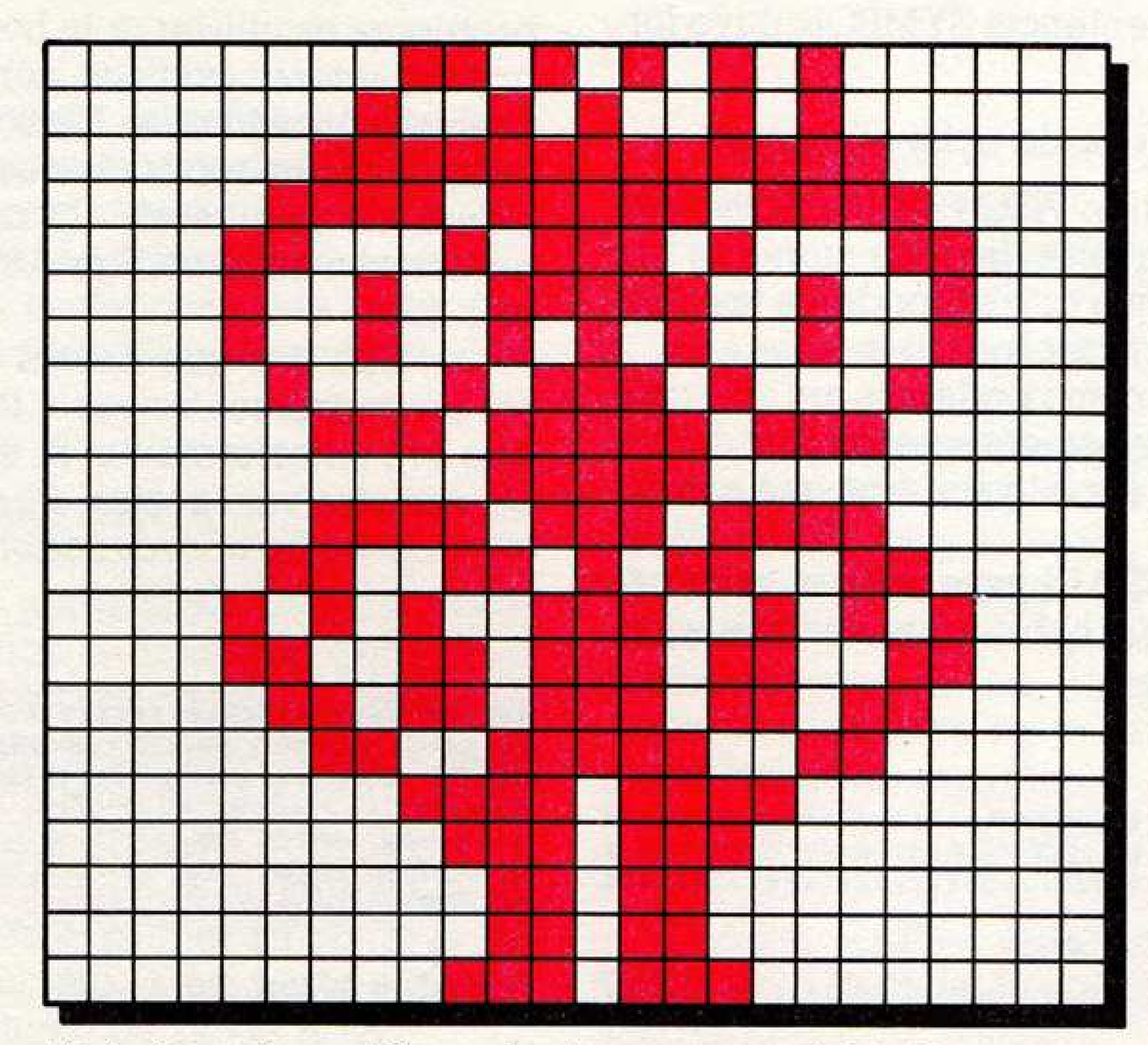


Fig. 5.—Los sprites en el Commodore tienen un tamaño de 24 × 21 puntos.

Este gracioso personaje (puede ser un búho, un marciano o cualquier otra cosa) está formado por 63 bytes. Para comprobarlo fíjate en la figura 6.

```
0 0 0 0 0 0 0 0   1 1 0 1 0 1 0 1   0 1 0 0 0 0 0 0   0, 213, 64
0 0 0 0 0 0 0 1   0 0 1 0 1 0 0 1   0 1 0 0 0 0 0 0   1, 41, 64
0 0 0 0 0 0 1 1   1 1 1 1 1 1 1 1   1 1 1 0 0 0 0 0   3, 255, 224
0 0 0 0 0 1 1 1   1 0 1 1 1 1 1 0   1 1 1 1 0 0 0 0   7, 190, 240
0 0 0 0 1 1 0 0   0 1 0 1 1 1 0 1   0 0 0 1 1 0 0 0   12, 93, 24
0 0 0 0 1 0 0 1   0 0 1 1 1 1 1 0   0 1 0 0 1 0 0 0   9, 62, 72
0 0 0 0 1 0 0 1   0 0 1 0 1 0 1 0   0 1 0 0 1 0 0 0   9, 42, 72
0 0 0 0 0 1 0 0   0 1 0 1 1 1 0 1   0 0 0 1 0 0 0 0   4, 93, 16
0 0 0 0 0 0 1 1   1 0 1 1 1 1 1 0   1 1 1 0 0 0 0 0   3, 190, 224
0 0 0 0 0 0 0 0   0 0 1 1 1 1 1 0   0 0 0 0 0 0 0 0   0, 62, 0
0 0 0 0 0 0 1 1   1 1 0 1 1 1 0 1   1 1 1 0 0 0 0 0   3, 221, 224
0 0 0 0 0 1 1 0   0 1 1 0 1 0 1 1   0 0 1 1 0 0 0 0   6, 107, 48
0 0 0 0 1 0 1 0   1 0 0 1 0 1 0 0   1 0 1 0 1 0 0 0   10, 148, 168
0 0 0 0 1 1 0 0   1 1 0 1 1 1 0 1   1 0 0 1 1 0 0 0   24, 221, 152
0 0 0 0 0 1 1 0   1 1 0 1 1 1 0 1   1 0 1 1 0 0 0 0   6, 221, 176
0 0 0 0 0 0 1 1   0 0 1 1 1 1 1 0   0 1 1 0 0 0 0 0   3, 62, 96
0 0 0 0 0 0 0 0   1 1 1 1 0 1 1 1   1 0 0 0 0 0 0 0   0, 247, 128
0 0 0 0 0 0 0 0   0 1 1 1 0 1 1 1   0 0 0 0 0 0 0 0   0, 119, 0
0 0 0 0 0 0 0 0   0 0 1 1 0 1 1 0   0 0 0 0 0 0 0 0   0, 54, 0
0 0 0 0 0 0 0 0   0 0 1 1 0 1 1 0   0 0 0 0 0 0 0 0   0, 54, 0
0 0 0 0 0 0 0 0   0 1 1 1 0 1 1 1   0 0 0 0 0 0 0 0   0, 119, 0
```

Fig. 6.—Así queda la malla con los 63 bytes al poner un cero donde hay blancos y un uno donde hay negros.

En cada fila de la malla hay tres BYTES y, como tenemos 21 filas, eso nos da 63 BYTES. Como siempre, donde hay un punto iluminado ponemos un uno y donde hay un hueco un cero. Como cada BYTE está en binario hay que pasarlo a base 10. Para hacerlo puedes utilizar el programa 1 y unirlo al programa 1 de esta misma sección cambiando las siguientes líneas:

```
 50 DIM N(63):LET BI=0
 60 FOR I=1 TO 63
170 FOR I=1 TO 63
```

Ahora ejecuta el programa 5 y verás el SPRITE en la pantalla.

```
10 REM ******************************************
20 REM * DEFINICION DE UN MUÑECO EN EL COMMODORE *
30 REM ******************************************
40 REM
50 PRINT "<SHIFT-HOME>"
60 V=53248
70 POKE V+21,1
80 POKE 2040,192
90 FOR I=0 TO 62
100    READ N
110    POKE 12288+I,N
120 NEXT I
130 POKE V+0,100
140 POKE V+1,100
150 END
200 REM
210 REM *** LINEAS DE DATAS ***
220 REM
230 DATA 0,213,64
240 DATA 1,41,64
250 DATA 3,255,224
260 DATA 7,190,240
270 DATA 12,93,24
280 DATA 9,62,72
290 DATA 9,42,72
300 DATA 4,93,16
310 DATA 3,190,224
320 DATA 0,62,0
330 DATA 3,221,224
340 DATA 6,107,48
350 DATA 10,148,168
360 DATA 24,221,152
370 DATA 6,221,176
380 DATA 3,62,96
390 DATA 0,247,128
400 DATA 0,119,0
410 DATA 0,54,0
420 DATA 0,54,0
430 DATA 0,119,0
```

La manera de definir SPRITES en el COMMODORE es bastante complicada, así que empezemos desde el principio.

El COMMODORE puede tener hasta 8 sprites numerados del 0 al 7. Estos SPRITES pueden ser de cuatro formas: normales, con ampliación en X, con ampliación en Y y con ampliación en X y en Y a la vez. Los SPRITES pueden trabajar en baja y en alta resolución indistintamente y podemos hacer que pasen unos por encima o por debajo de otros.

Para definir los SPRITES nos vamos a valer de 46 registros que tiene el COMMODORE para su control. Estos registros están descritos en la figura 7.

De todos los registros que hay en la ta-

Reg. N.°	FUNCION
0	Coordenada X del Sprite 0
1	Coordenada Y del Sprite 0
2	Coordenada X del Sprite 1
3	Coordenada Y del Sprite 1
4	Coordenada X del Sprite 2
5	Coordenada Y del Sprite 2
6	Coordenada X del Sprite 3
7	Coordenada Y del Sprite 3
8	Coordenada X del Sprite 4
9	Coordenada Y del Sprite 4
10	Coordenada X del Sprite 5
11	Coordenada Y del Sprite 5
12	Coordenada X del Sprite 6
13	Coordenada Y del Sprite 6
14	Coordenada X del Sprite 7
15	Coordenada Y del Sprite 7
16	Coordenada acta en X
17	Scroll/Modo
18	Rastreo
19	Lápiz óptico en X
20	Lápiz óptico en Y
21	Visualización de Sprites
22	Scroll/Modo
23	Ampliación en Y del Sprite
24	Memoria de carácter de pantalla
25	Interruptor/Demanda
26	Interruptor/Demanda
27	Prioridad de fondo del Sprite
28	Selector de color del Sprite
29	Ampliación en X
30	Choque de Sprites
31	Choque de Fondo y Sprite
32–46	COLORES LOS SPRITES

Fig. 7.—Tabla de los registros para usar los sprites en el Commodore.

bla, de momento sólo hablaremos de algunos de ellos. Esperaremos a saber más cosas para referirnos al resto.

Los registros 0 y 1 le dicen al ordenador en qué lugar de la pantalla aparecerá el SPRITE número 0 cuando esté definido. Los registros 2 y 3 hacen lo mismo, pero con el SPRITE número 1. Como se puede apreciar, los 14 primeros registros van emparejados de dos en dos, actuando cada par sobre un SPRITE determinado.

El número 21 le dice al ordenador si un SPRITE dado tiene que estar visible u oculto en la pantalla.

El 23 nos indica si un determinado SPRITE tiene altura normal o ampliada. La altura ampliada es el doble de la normal.

El último registro que nos interesa de momento es el 29, cuya función es la de indicar si un SPRITE tiene anchura normal o ampliada (doble).

El resto de los registros serán estudiados según vayamos avanzando y comprendiendo cómo se realiza la animación.

Si nos fijamos en el programa 5 lo primero que hacemos es dar a la variable V el valor 53248. Esta es la dirección donde empiezan los registros del COMMODORE. Según esto, la dirección 53248 es el registro 0, la 53249 el registro 1, la 53250 el registro 3 y así hasta la dirección 53294, que sería el registro 46.

En la línea siguiente pokeamos en la dirección V+21 el número 1. Si nos fijamos en la tabla, el registro 21 es el que nos dice si un SPRITE está, en un momento dado, visible en la pantalla. Como sólo tenemos un registro y, sin embargo, tenemos ocho SPRITES posibles, esto significa que cada BIT del registro nos indica el atributo (encendido=visible, apagado=invisible) de cada SPRITE. Si el BIT 0 del registro está puesto a 1 esto significa que el SPRITE número 0 será visible. Si el que está puesto a uno es el BIT 4, significará que será visible el SPRITE número 4. Si se encuentran encendidos los BITs 2 y 4 significará que serán visibles los SPRITES 2 y 4. Si están todos los BITs a 1 se verán todos los SPRITES.

Para entender mejor todo esto fíjate en la figura 8.

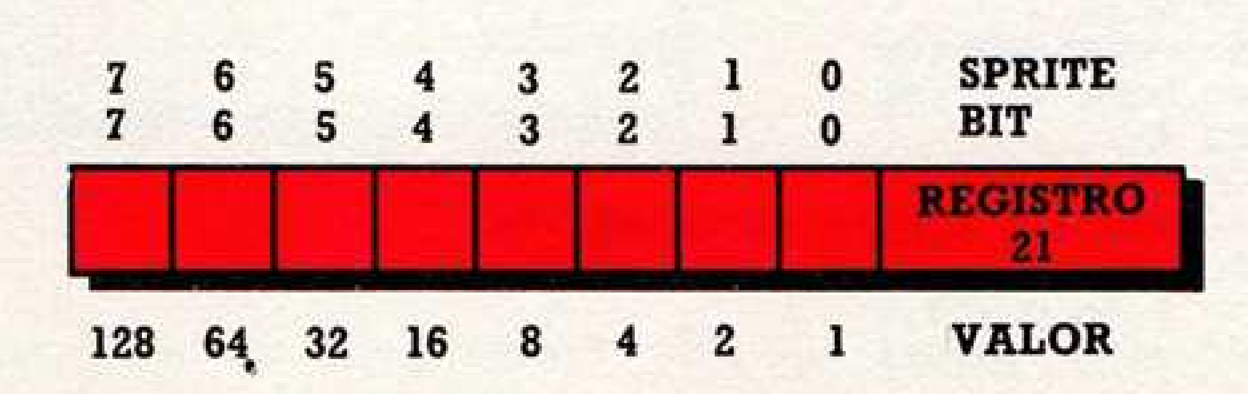


Fig. 8.—Uso de los bits del registro 21.

Según esto, cuando ponemos 'POKE V+21,1' lo que estamos haciendo es poner a 1 el BIT número cero del registro 21. Esto significa que se hará visible el SPRITE número 0.

En la línea 80 pokeamos en la dirección 2040 un 192. Esto le dice al COMMODORE que la definición de los SPRITES comienza en el área 192 de memoria. El significado de esto no es importante para realizar SPRITES, por lo que lo dejaremos para más adelante.

A partir de la línea 90, y hasta la 120, se realiza un bucle cuya función es leer la definición del SPRITE de las líneas de DATA y pokear a partir de la dirección 12288. Esta es la dirección donde empieza el área 192 de memoria. Si necesitamos definir otro SPRITE haremos que éste sea el número 1, con lo que, en el registro 21, pokearemos lo que haya en el OR 2.

POKE V+21, PEEK(V+21) OR 2

Haciendo esta operación se consigue que si el BIT 1 no está puesto a 1 se encienda. En el caso que estuviese puesto a 1 así se conservaría. Ya veremos esto con más profundidad cuando estudiemos las operaciones lógicas.

Para decir en qué área se encuentra la definición de este nuevo SPRITE pokearemos en la dirección 2041 un 193, que es la siguiente área de memoria, e introduciremos los datos del SPRITE a partir de la dirección 12352.

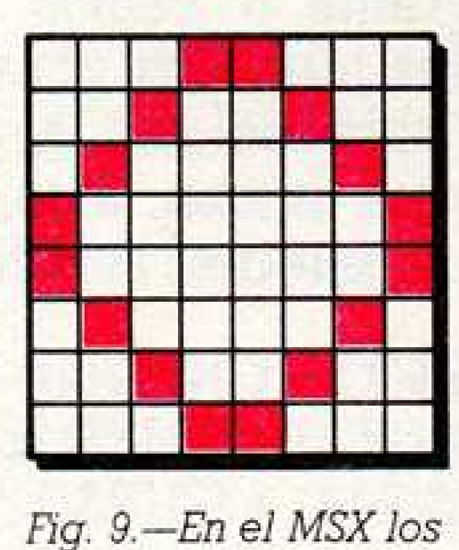
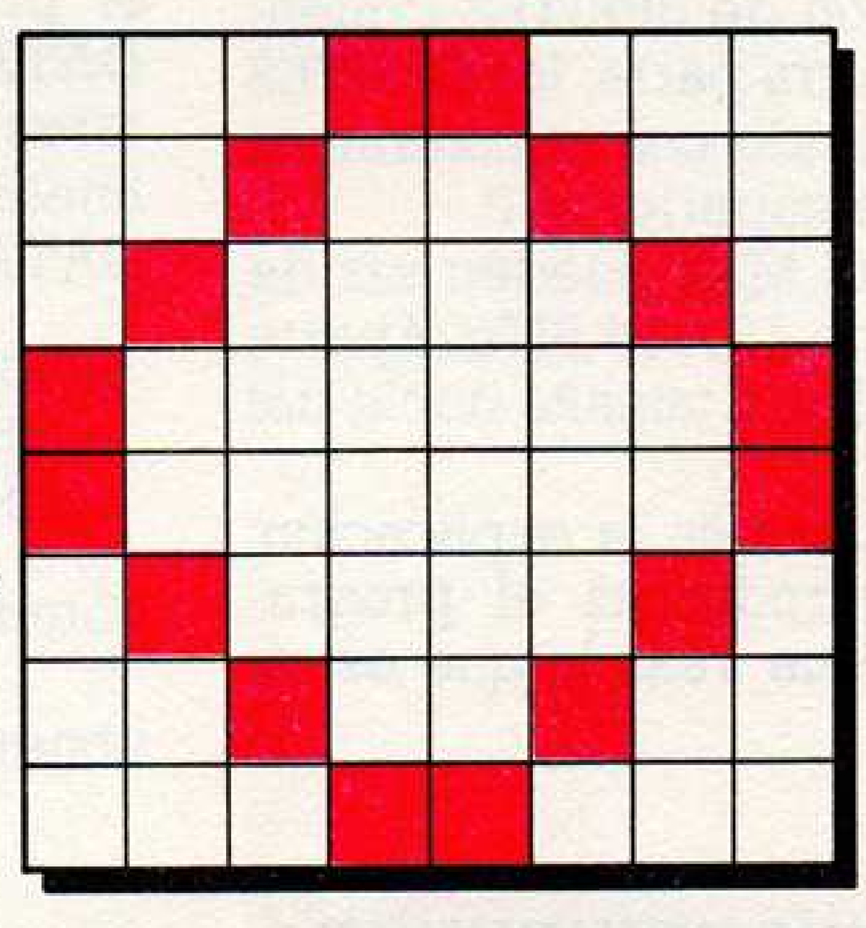


Fig. 9.—En el MSX los sprites pueden ampliarse y hacer que tengan el doble de tamaño.



Aunque todo esto parezca muy complicado no lo es. Cada SPRITE tiene que estar en un área de memoria. Cada área de memoria ocupa 64 BYTES. Según esto, si el SPRITE número 0 está en el área 192, el número 1 estará en la 193, el número 2 en la 194 y así sucesivamente. El área de memoria 192 empieza en la dirección 12288, el área 193 en 12288 + 64 = 12352, la número 194 en 12288 + 2*64 y así sucesivamente.

Con todo lo dicho queda claro cómo se definen los SPRITES. Pero ¿cómo ponerlos en la pantalla? Si nos fijamos en la tabla de la figura 7, los registros 0 y 1 nos dicen la posición de dicho SPRITE en la pantalla. Pokeando en dichos registros un número entre 0 y 255 el SPRITE aparecerá ante nuestros ojos (líneas 130 y 140 del programa 5). Por supuesto, si definimos el SPRITE número 1, las coordenadas de dicho SPRITE en la pantalla tenemos que introducirlas en los registros 2 y 3. Si definimos el número 2, sus coordenadas habrá que ponerlas en los registros 4 y 5 y así sucesivamente.

Todo esto es un poco complicado. La mejor manera de entenderlo del todo es definiendo vuestros propios SPRITES, probando con diferentes áreas de memoria (siempre mayores o iguales a 192) y colocándolos en diferentes lugares de la pantalla.

Antes hemos comentado que los SPRITES en el COMMODORE pueden ampliarse en X, en Y e incluso en ambas direcciones a la vez.

Para ampliar un SPRITE en la dirección X utilizaremos el registro número 29. Su modo de funcionar es igual que el modo del registro 21. Como tenemos ocho SPRITES y un solo BYTE para definir si su anchura es normal o ampliada, a cada SPRITE le corresponde un BIT de dicho BYTE. Según esto, si el BIT número 3 está a 1 significa que el SPRITE núme-

ro 3 está ampliado. Si estuviesen a 1 los BITs 2, 4 y 7 significaría que estarían ampliados los SPRITES 2, 4 y 7.

Para ampliar los SPRITES en la dirección Y se utiliza el registro número 20. Su utilización es igual que la del 29.

Para hacer ampliaciones en las dos direcciones (X e Y) se ponen a 1 los BITs correspondientes de los dos registros (el 20 y el 26).

Definición de caracteres en el MSX

En el MSX es realmente complicado definir caracteres nuevos y además sólo se puede realizar en muy contadas ocasiones. A cambio, permite la definición de SPRITES ('duendecillo' en inglés). Por otra parte, los SPRITES sólo pueden usarse cuando nos encontramos en un modo de pantalla superior al 0.

Los SPRITES en el MSX pueden ser de dos tamaños: 8 × 8 BITs o 16 × 16 BITs. Aparte de esto se pueden ampliar a tamaño doble del normal.

Antes de continuar con la explicación, te recomiendo que introduzcas el programa y lo ejecutes para que veas lo que es un SPRITE.

```
10 REM *************************************
20 REM * DEFINICION DE UN SPRITE EN EL MSX *
30 REM *************************************
40 REM
50 FOR I=1 TO 8
60    READ A
70    A$=A$+CHR$(A)
80 NEXT I
90 SPRITE$(1)=A$
100 REM
110 REM *** POSICIONAMIENTO DEL SPRITE ***
120 REM
130 SCREEN 1
140 PUT SPRITE 1,(10,10),2,1
150 GOTO 150
160 REM
170 REM *** LINEA DE DATAS ***
180 REM
190 DATA 60,126,255,255,253,249,122,60
```

En este programa hemos definido la pelotita de la figura 1. Como vimos anteriormente, después de definir el dibujo hay que sustituir los puntos en negro por unos y los puntos en blanco por ceros. Como la cuadrícula es de 8 × 8 puntos esto nos da ocho números escritos en base 2 unos debajo de otros. Para pasar estos números en base 2 a base 10 puedes utilizar el programa 1 uniéndolo al programa 1 de la sección de TRUCOS DE PROGRAMACION.

Una vez obtenidos dichos números en base 10 hay que introducirlos, de alguna manera, dentro del ordenador de forma que él sepa que estamos definiendo un SPRITE. La forma de hacerlo es mediante la sentencia:

SPRITE$(n) = cadena alfanumérica

Si a cada fila de la figura definida la denominamos como f1, f2, f3... f8, el significado de CADENA ALFANUMERICA es:

CHR$(f1) + CHR$(f2) + CHR$(f3) + + ... + CHR$(f8)

De esta forma se entiende por qué en el programa 6 vamos leyendo de la línea DATA los números convirtiéndolos en caracteres y uniéndolos a la variable A$. Todas las líneas que hacen esto, incluyendo la línea DATA, se pueden sustituir por una sola línea:

```
SPRITE$(1) = CHR$(60) + CHR$(126) +
+ CHR$(255) + CHR$(255) + CHR$(253) +
+ CHR$(249) + CHR$(122) + CHR$(60)
```

obteniéndose el mismo resultado.

Una vez definido el SPRITE podemos ponerlo en la pantalla mediante la instrucción:

PUT SPRITE np,(X,Y)c,ns

donde 'np' es el número de plano (hay 32 planos distintos y cada SPRITE tiene que estar en un plano), X e Y son las coordenadas de la pantalla donde queremos que aparezca el SPRITE, 'c' es el color del SPRITE (un número entre 0 y 15) y 'ns' es el número de SPRITE.

La forma de definir SPRITES de 16 × 16 puntos así como la definición de plano serán vistas en capítulos posteriores.

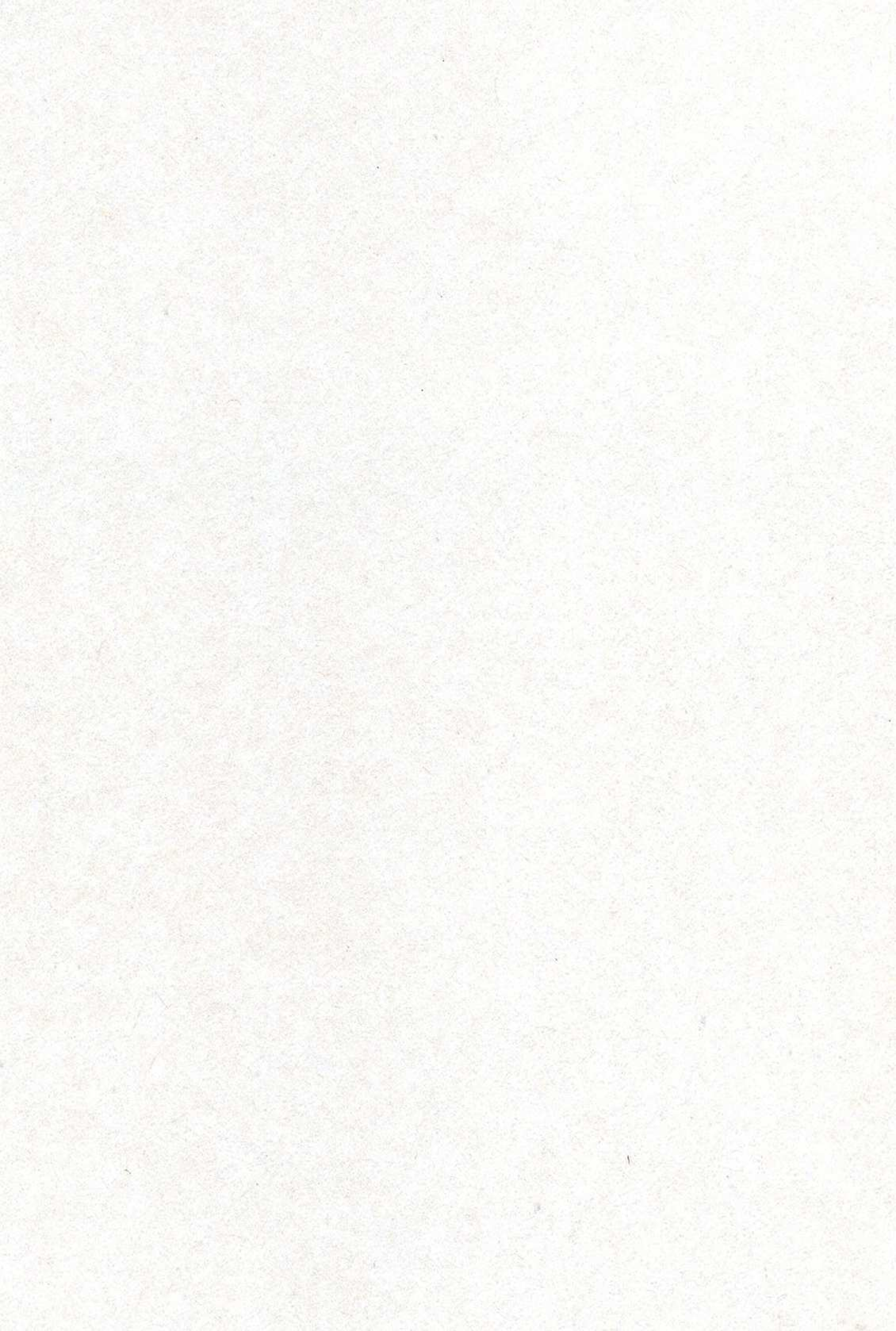
Te recomiendo que pruebes tú mismo a definir tus propios SPRITES para que te vayas acostumbrando. Como ejemplo puedes definir los que te muestro en la figura 4, aunque te recomiendo que también dibujes los que a ti se te ocurran.

Mas arriba hemos dicho que el MSX ofrece la posibilidad de ampliar el tamaño de los SPRITES. La forma de realizarlo es sencillísima. Basta con ponerle un nuevo parámetro a la sentencia SCREEN cuando abrimos la pantalla. La sintaxis es:

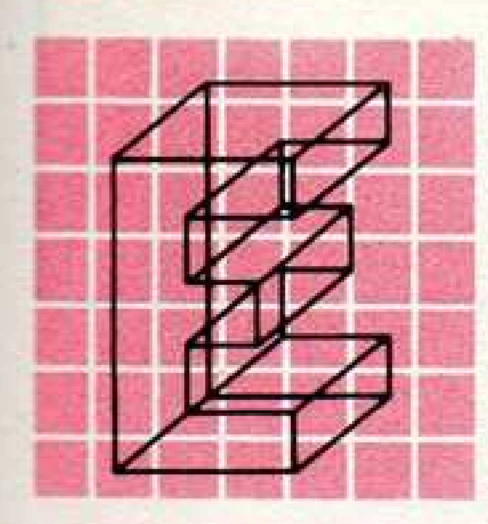
SCREEN p,n

Donde 'p' es el modo de pantalla (siempre mayor que cero) y 'n' es la forma de los SPRITES. Según el valor de 'n' los SPRITES serán:

n = 0 → 8 × 8 sin ampliar
n = 1 → 8 × 8 ampliado
n = 2 → 16 × 16 sin ampliar
n = 3 → 16 × 16 ampliado



Cambio de bases

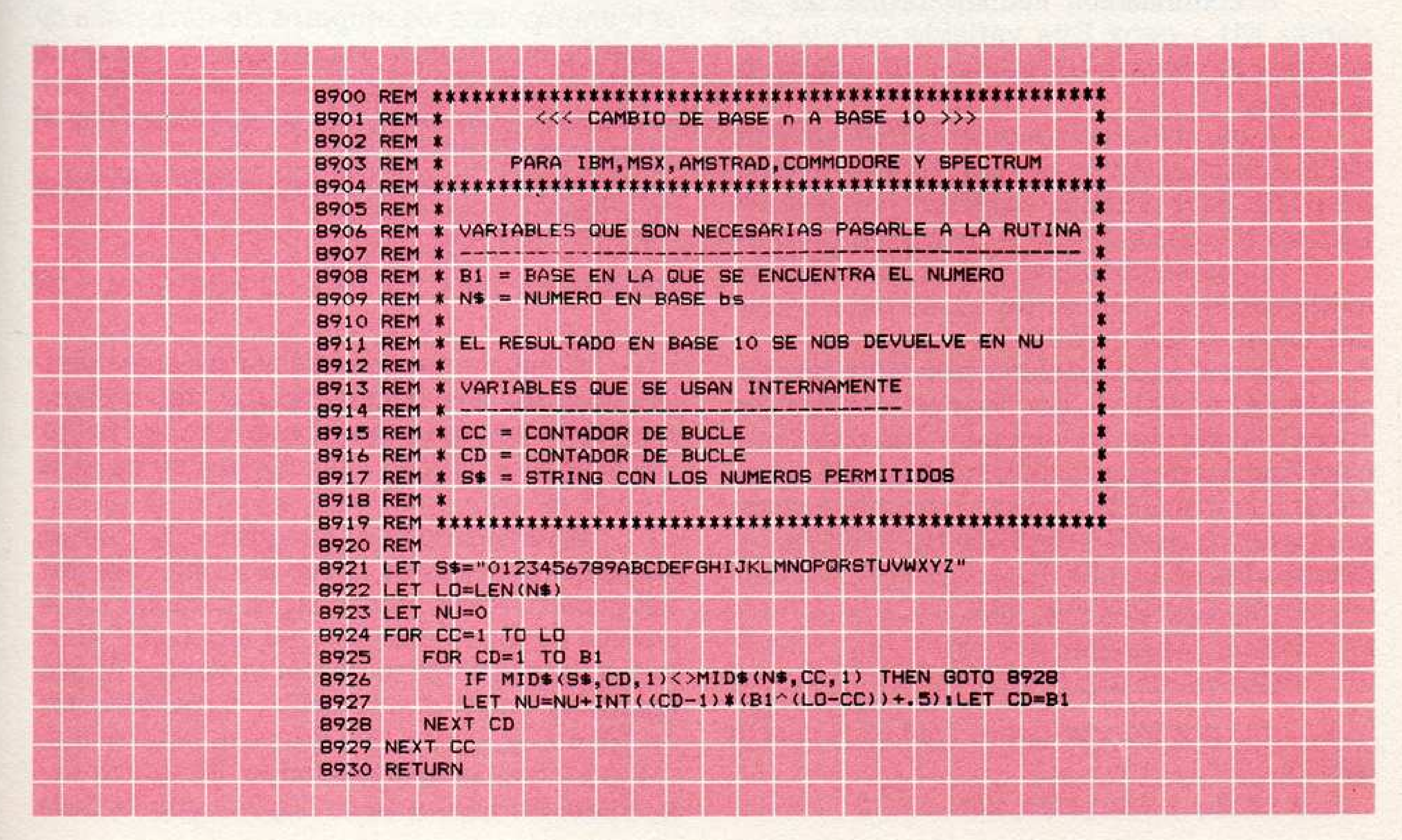


EN el mundo de la informática es común trabajar en bases distintas a la base 10, que es con la que nosotros estamos acostumbrados a manejar. Los hombres trabajamos en base 10 porque tenemos diez dedos entre las dos manos que nos permiten contar, pero el ordenador sólo sabe contar con un dedo, con 8 dedos y con 16 dedos. La razón de esto se escapa del propósito de esta sección, aunque siempre es interesante saber contar como el ordenador para así poder hacer de nuestros programas una pequeña obra de arte.

Los programas que proponemos a continuación nos permitirán pasar un número que esté en cualquier base a base 10, pasar un número que esté en base 10 a otra base, pasar un número de una base a otra y hacer operaciones en cualquier base.

Así pues, y sin más demora, entramos en el primer programa.

Cambio de base b1 a base 10



```
8900 REM ****************************************************
8901 REM *       <<< CAMBIO DE BASE n A BASE 10 >>>          *
8902 REM *                                                   *
8903 REM *     PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM     *
8904 REM ****************************************************
8905 REM *                                                   *
8906 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA  *
8907 REM * ------------------------------------------------- *
8908 REM * B1 = BASE EN LA QUE SE ENCUENTRA EL NUMERO        *
8909 REM * N$ = NUMERO EN BASE bs                            *
8910 REM *                                                   *
8911 REM * EL RESULTADO EN BASE 10 SE NOS DEVUELVE EN NU     *
8912 REM *                                                   *
8913 REM * VARIABLES QUE SE USAN INTERNAMENTE                *
8914 REM * ----------------------------------                *
8915 REM * CC = CONTADOR DE BUCLE                            *
8916 REM * CD = CONTADOR DE BUCLE                            *
8917 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS            *
8918 REM *                                                   *
8919 REM ****************************************************
8920 REM
8921 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8922 LET LO=LEN(N$)
8923 LET NU=0
8924 FOR CC=1 TO LO
8925    FOR CD=1 TO B1
8926       IF MID$(S$,CD,1)<>MID$(N$,CC,1) THEN GOTO 8928
8927       LET NU=NU+INT((CD-1)*(B1^(LO-CC))+.5):LET CD=B1
8928    NEXT CD
8929 NEXT CC
8930 RETURN
```

Este primer programa nos permite pasar un número que se encuentre en cualquier base (menor o igual que 36) a base 10.

Los parámetros que tenemos que pasar a la rutina son los siguientes:

En N$ pasamos el número que queremos pasar a base 10.

En B1 pasamos la base en la que se encuentre el número.

Cuando la rutina retorne de su ejecución, nos devolverá el número N$ que estaba en base B1, en la variable NU.

El programa funciona de la siguiente manera:

En la línea 8921 definimos la variable S$ con todos los números y las letras del abecedario en mayúsculas. Si te fijas bien el uno (1) está en posición de dos, el nueve (9) en la décima, la F en la decimoséptima, etc. Esto significa que cada letra o número está un lugar más a la derecha del que le correspondería si los numerásemos del 1 al 36. Ya veremos qué pasa con esto.

En la línea 8922 hacemos que la variable 'LO' tome la longitud de N$. El número se encuentra en una variable alfanumérica porque cuando pasamos a contar en una base mayor que 10, como no tenemos números suficientes, se utilizan las letras. Así el número FFFF en hexadecimal (base 16), no puede almacenarse en una variable numérica.

A continuación inicializaremos las variables NU a cero. Esta variable será la que nos dirá el número N$ en base 10 al final de la rutina.

A partir de este momento comienza un bucle enlazado (un bucle dentro de otro) cuyo propósito es ir mirando carácter a carácter el número N$, pasando dicho número a base 10.

Antes de continuar vamos a dar algo de teoría sobre el cambio de bases para entender mejor el programa.

Cuando contamos en base 10 y nos pasamos del número 9, por ejemplo cuando decimos 132, lo que hacemos es: cada vez que nos pasamos de 9, nos llevamos una y la apuntamos a la izquierda. Lo mismo ocurre cuando nos pasamos de 99 y de 999 y de 9999. Así, el número 132 es igual a:

Una vez cien (1×100)
Tres veces diez (3×10)
Dos veces uno (2×1)

Y si nos fijamos, poner esto es lo mismo que:

$$132 = 1 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$$

Esto es igual para todas las bases. Si tenemos el número 101001 en base 2 en base 10 sería igual a:

$$101001 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^4 + 0 \times 2^2 + \\ + 0 \times 2^1 + 1 \times 2^0 = 41$$

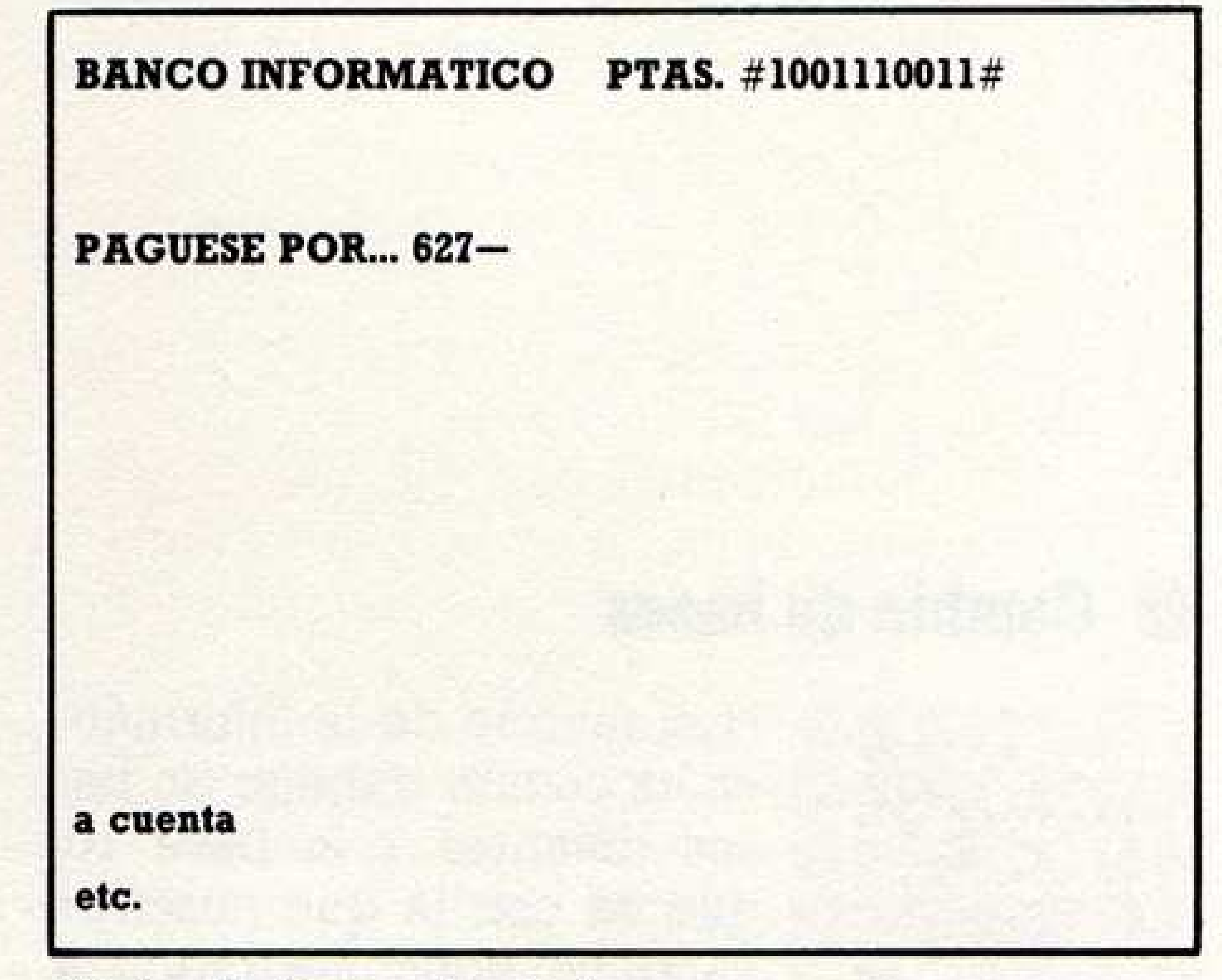


Fig. 1.—¿Te da para el coche?

Si te fijas bien, los exponentes que colocamos después del 2 van decreciendo en uno cada vez que cogemos un número que está a la derecha del anterior. Por otra parte, si numerásemos los números de derecha a izquierda, empezando por el cero (0), nos daría el exponente. Para dejar las cosas claras veamos un ejemplo más antes de continuar con el programa.

Vamos a pasar el número 32FA en base 16 a base 10:

$$32FA = 3 \times 16^3 + 2 \times 16^2 + 15 \times 16^1 + 10 \times 16^0$$

Como se puede apreciar en este ejemplo, cuando nos hemos pasado del número nueve (9) empezamos a contar con las letras. Así la letra A tiene el valor 10, la F valor 15, la Z valor 34, etc.

Siguiendo con el programa, lo que va haciendo el bucle es mirar si algún carácter del número N$ es igual a algún carácter de la cadena S$. En el caso de que alguno de ellos lo sea, entonces hace que la variable NU sea igual a la posición que ocupa dicho carácter de N$ en S$ menos uno (por eso se hizo hincapié en que la numeración en S$ estaba aumentada en uno) y se multiplica por la base

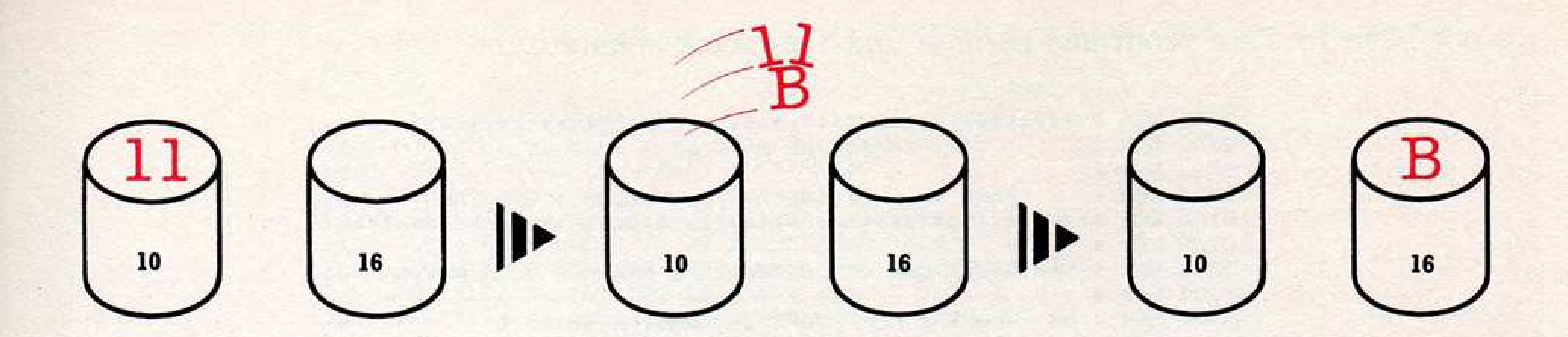
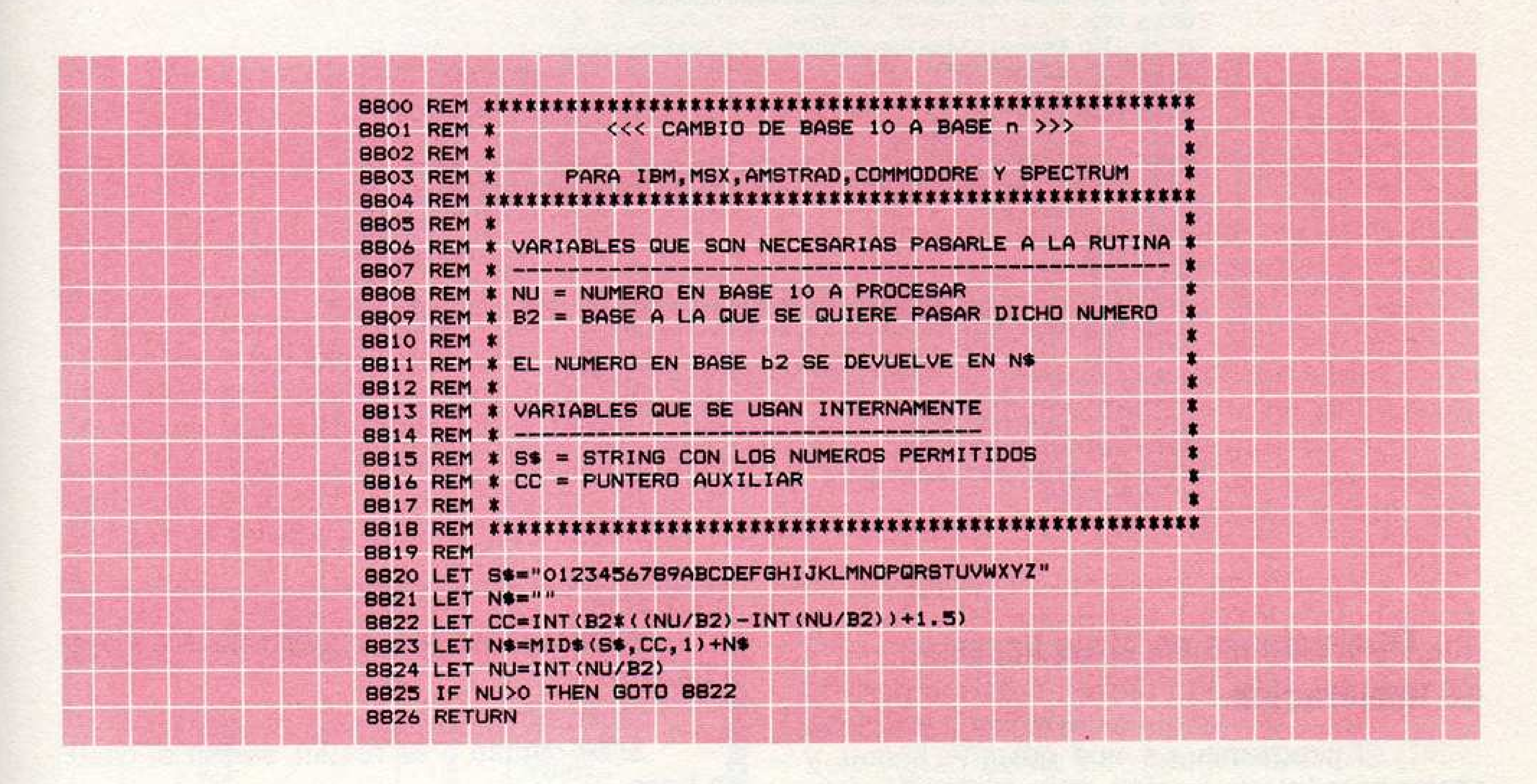


Fig. 2.—Al pasar de una base a otra, aunque cambien, los dígitos no cambian su valor.

elevada a la posición del carácter del número N$ menos uno.

En las siguientes pasadas que dé el bucle, NU tendrá un valor, por lo que se suma, al antiguo valor, el que sacamos al mirar el dígito siguiente. El bucle se repite hasta que se miran todos los dígitos del número.

Cambio de base 10 a base B2



```
8800 REM ***************************************************
8801 REM *         <<< CAMBIO DE BASE 10 A BASE n >>>       *
8802 REM *                                                 *
8803 REM *     PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM   *
8804 REM ***************************************************
8805 REM *                                                 *
8806 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
8807 REM * ----------------------------------------------- *
8808 REM * NU = NUMERO EN BASE 10 A PROCESAR               *
8809 REM * B2 = BASE A LA QUE SE QUIERE PASAR DICHO NUMERO *
8810 REM *                                                 *
8811 REM * EL NUMERO EN BASE b2 SE DEVUELVE EN N$          *
8812 REM *                                                 *
8813 REM * VARIABLES QUE SE USAN INTERNAMENTE              *
8814 REM * --------------------------------                *
8815 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS          *
8816 REM * CC = PUNTERO AUXILIAR                           *
8817 REM *                                                 *
8818 REM ***************************************************
8819 REM
8820 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8821 LET N$=""
8822 LET CC=INT(B2*((NU/B2)-INT(NU/B2))+1.5)
8823 LET N$=MID$(S$,CC,1)+N$
8824 LET NU=INT(NU/B2)
8825 IF NU>0 THEN GOTO 8822
8826 RETURN
```

El paso de un número de base diez (10) a otra base es más sencillo que el inverso. Lo único que hay que hacer es dividir el número en base 10 por la nueva base hasta que el resto nos de 0 o sea imposible hacer la división exacta. Así, si queremos pasar el número 322 en base 10 a base 5, haremos lo siguiente:

```
322  |5
 22   64  |5
 (2)  14   12  |5
      (4)  (2)  (2)
```

Una vez hemos hecho esto cogemos el último número que nos haya salido, que ya no sea divisible por la nueva base, y lo ponemos seguido de los restos que nos haya dado empezando por el último. De esta manera el número 322 en base 10 es igual al número 2242 en base 5.

Lo que hace el programa 2.2 es justamente lo que acabamos de ver. La variable CC nos irá dando la posición dentro de S$, que tenemos que coger para luego almacenarla en la variable N$. N$ nos retorna el número NU en base B2.

Con todo lo que hemos visto, podemos hacer un programa que nos permita pasar un número en una base B1 a otra base B2 pasan-

do por base 10. Este programa sería la unión de los dos anteriores.

```
8700 REM ***************************************************
8701 REM *          <<< CAMBIO DE BASE b1 A BASE b2 >>>       *
8702 REM *                                                    *
8703 REM *     PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM      *
8704 REM ***************************************************
8705 REM *                                                    *
8706 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA   *
8707 REM * ------------------------------------------------   *
8708 REM * N$ = NUMERO QUE QUEREMOS CAMBIAR DE BASE           *
8709 REM * B1 = BASE EN LA QUE SE ENCUENTRA EL NUMERO         *
8710 REM * B2 = BASE A LA QUE QUEREMOS PASAR EL NUMERO        *
8711 REM *                                                    *
8712 REM * EL RESULTADO SE NOS DEVUELVE EN N$                 *
8713 REM *                                                    *
8714 REM * VARIABLES QUE SE USAN INTERNAMENTE                 *
8715 REM * ---------------------------------                  *
8716 REM * CC = CONTADOR DE BUCLES Y PUNTERO AUXILIAR         *
8717 REM * CD = CONTADOR DE BUCLES                            *
8718 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS             *
8719 REM *                                                    *
8720 REM ***************************************************
8721 REM
8722 REM =================================
8723 REM ==  PASO DE BASE b1 A BASE 10  ==
8724 REM =================================
8725 REM
8726 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8727 LET LO=LEN(N$)
8728 LET NU=0
8729 FOR CC=1 TO LO
8730    FOR CD=1 TO B1
8731       IF MID$(S$,CD,1)<>MID$(N$,CC,1) THEN GOTO 8733
8732       LET NU=NU+INT((CD-1)*(B1^(LO-CC))+.5):LET CD=B1
8733    NEXT CD
8734 NEXT CC
8735 REM
8736 REM =================================
8737 REM ==  PASO DE BASE 10 A BASE b2  ==
8738 REM =================================
8739 REM
8740 LET N$=""
8741 LET CC=INT(B2*((NU/B2)-INT(NU/B2))+1.5)
8742 LET N$=MID$(S$,CC,1)+N$
8743 LET NU=INT(NU/B2)
8744 IF NU>0 THEN GOTO 8741
8745 RETURN
```

Operaciones en otras bases de numeración

El programa 2.4 nos permite sumar y restar números que estén en una base B1 y obtener el resultado en dicha base y en base 10.

La forma de trabajar este programa es le siguiente:

1. Se pasan los dos números (N1 y N2) a base 10.

2. Se suman o se restan, según el caso, en base 10.

3. Se vuelven a pasar a la base original B1.

```
8600 REM ***************************************************
8601 REM *      <<< SUMA Y RESTA EN UNA BASE CUALQUIERA >>>   *
8602 REM *                                                    *
8603 REM *     PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM      *
8604 REM ***************************************************
8605 REM *                                                    *
8606 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA   *
8607 REM * ------------------------------------------------   *
8608 REM * P$ = PRIMER OPERANDO                               *
8609 REM * Q$ = SEGUNDO OPERANDO                              *
8610 REM * B1 = BASE EN LA QUE ESTAN LOS OPERANDOS            *
8611 REM * SW = TIPO DE OPERACION                             *
8612 REM *       SW=1 --> SUMA.   P$+Q$                       *
```

```
8613 REM *       SW=0 --> RESTA. P$-Q$                        *
8614 REM *                                                    *
8615 REM *   EL RESULTADO SE DEVUELVE EN T$ EN BASE B1 Y EN   *
8616 REM * TT EN BASE 10                                      *
8617 REM *                                                    *
8618 REM * VARIABLES USADAS INTERNAMENTE                      *
8619 REM * -----------------------------                      *
8620 REM * N$ = VARIABLE AUXILIAR                             *
8621 REM * LO = LONGITUD DE N$ EN CARACTERES                  *
8622 REM * CC = CONTADOR DE BUCLE Y PUNTERO AUXILIAR          *
8623 REM * CD = CONTADOR DE BUCLE                             *
8624 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS             *
8625 REM * NU = VARIABLE AUXILIAR                             *
8626 REM *                                                    *
8627 REM ******************************************************
8628 REM
8629 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8630 N$=P$
8631 GOSUB 8640
8632 N1=NU
8633 N$=Q$
8634 GOSUB 8640
8635 N2=NU
8636 IF SW=1 THEN LET TT=N1+N2:GOTO 8638
8637 TT=N1-N2
8638 GOSUB 8652
8639 RETURN
8640 REM
8641 REM *** CAMBIO DE BASE B1 A BASE 10 ***
8642 REM
8643 LET LO=LEN(N$)
8644 LET NU=0
8645 FOR CC=1 TO LO
8646    FOR CD=1 TO B1
8647       IF MID$(S$,CD,1)<>MID$(N$,CC,1) THEN GOTO 8649
8648       LET NU=NU+INT((CD-1)*(B1^(LO-CC))+.5):LET CD=B1
8649    NEXT CD
8650 NEXT CC
8651 RETURN
8652 REM
8653 REM *** PASO DE BASE 10 A BASE B1 ***
8654 REM
8655 LET T$=""
8656 LET CC=INT(B1*((TT/B1)-INT(TT/B1))+1.5)
8657 LET T$=MID$(S$,CC,1)+T$
8658 LET TT=INT(TT/B1)
8659 IF TT>0 THEN GOTO 8656
8660 RETURN
```

DIGITO	VALOR NUMERICO	DIGITO	VALOR NUMERICO
0	CERO	I	DIECIOCHO
1	UNO	J	DIECINUEVE
2	DOS	K	VEINTE
3	TRES	L	VEINTIUNO
4	CUATRO	M	VEINTIDOS
5	CINCO	N	VEINTITRES
6	SEIS	O	VEINTICUATRO
7	SIETE	P	VEINTICINCO
8	OCHO	Q	VEINTISEIS
9	NUEVE	R	VEINTISIETE
A	DIEZ	S	VEINTIOCHO
B	ONCE	T	VEINTINUEVE
C	DOCE	U	TREINTA
D	TRECE	V	TREINTA Y UNO
E	CATORCE	W	TREINTA Y DOS
F	QUINCE	X	TREINTA Y TRES
G	DIECISEIS	Y	TREINTA Y CUATRO
H	DIECISIETE	Z	TREINTA Y CINCO

Fig. 3.—Estos son los valores de cada digito.

NOTA:

Debido a que el SPECTRUM no tiene la función LEFT$ ni RIGHT$ ni MID$, para simularlo hay que hacer lo siguiente:

LEFT$ (A$, I) se pondrá como A$ (TO I)

RIGHT$ (A$, I) se pondrá como A$ (I TO)

MID$ (A$, I, J) se pondrá como A$ (I TO I+J). En el caso especial en que J fuese igual a uno (1) se pondría A$ (I).

Por ejemplo, en el programa 2.2 en la línea 8926 pone:

```
8926 IF MID$ (S$, CD, 1) <> MID$ (N$, CC, 1)
     THEN GOTO 8928
```

los usuario del SPECTRUM tendrían que poner:

```
8926 IF S$ (CD) <> N$ (CC) THEN GOTO 8928
```

para que les funcione.

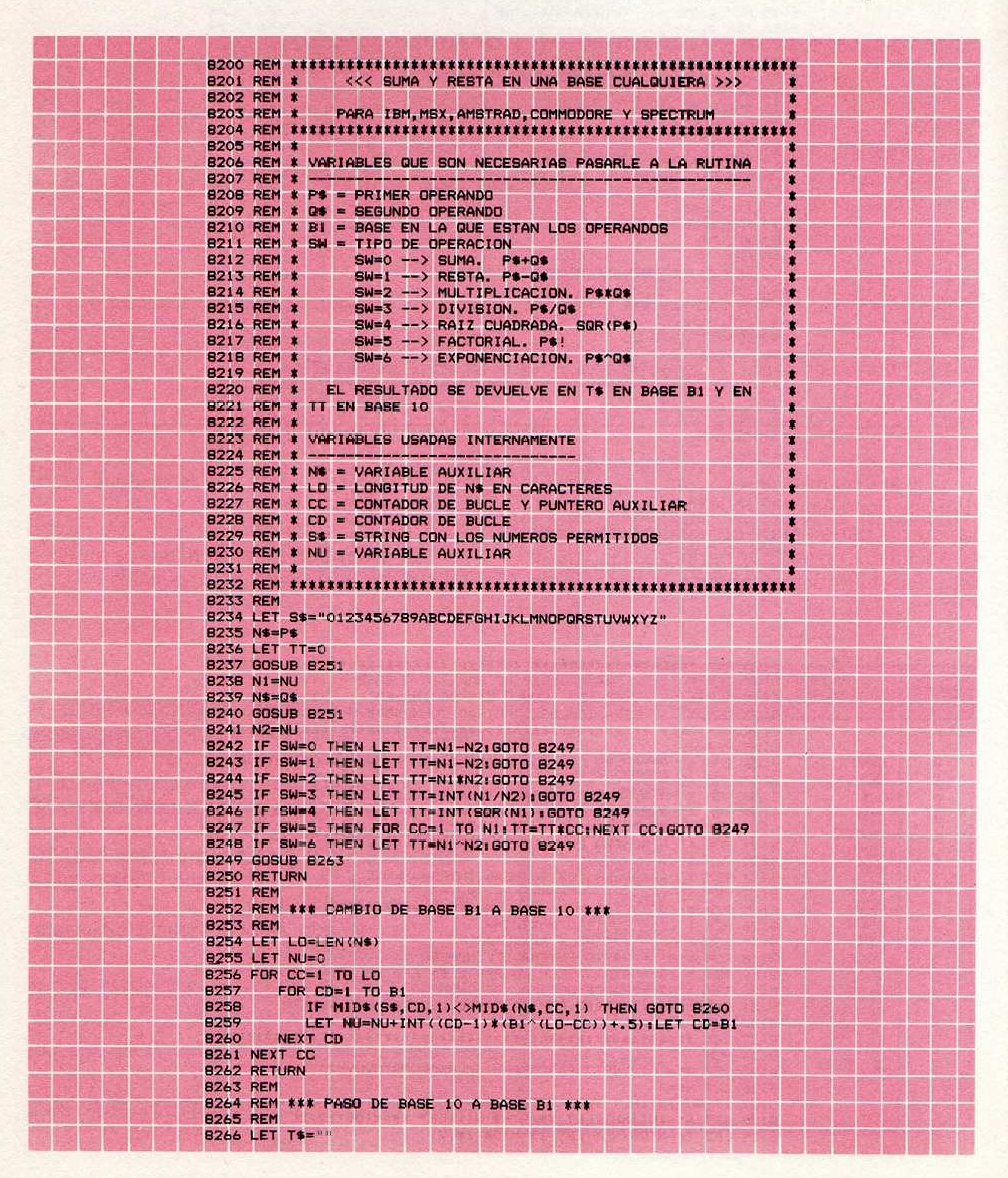
Ejercicios resueltos

EJERCICIO N.º 1

¿Podrías hacer un programa que multiplicase, dividiese, hiciese raíces cuadradas, elevase a una potencia, hallase el factorial, etcétera, en distintas bases?

SOLUCION

Como hemos dicho un poco más arriba, la mejor manera de hacer operaciones con números que se encuentren en bases distintas de la 10 es pasar dichos números a base 10, operar y volverlos a pasar a su base original. El programa 2.5 es muy parecido al 2.4, pero con la diferencia de que realiza más operaciones.



```
8200 REM ***************************************************
8201 REM *     <<< SUMA Y RESTA EN UNA BASE CUALQUIERA >>>     *
8202 REM *                                                     *
8203 REM *    PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM        *
8204 REM ***************************************************
8205 REM *                                                     *
8206 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA    *
8207 REM * ------------------------------------------------    *
8208 REM * P$ = PRIMER OPERANDO                                *
8209 REM * Q$ = SEGUNDO OPERANDO                               *
8210 REM * B1 = BASE EN LA QUE ESTAN LOS OPERANDOS             *
8211 REM * SW = TIPO DE OPERACION                              *
8212 REM *       SW=0 --> SUMA.  P$+Q$                         *
8213 REM *       SW=1 --> RESTA. P$-Q$                         *
8214 REM *       SW=2 --> MULTIPLICACION. P$*Q$                *
8215 REM *       SW=3 --> DIVISION. P$/Q$                      *
8216 REM *       SW=4 --> RAIZ CUADRADA. SQR(P$)               *
8217 REM *       SW=5 --> FACTORIAL. P$!                       *
8218 REM *       SW=6 --> EXPONENCIACION. P$^Q$                *
8219 REM *                                                     *
8220 REM *   EL RESULTADO SE DEVUELVE EN T$ EN BASE B1 Y EN    *
8221 REM * TT EN BASE 10                                       *
8222 REM *                                                     *
8223 REM * VARIABLES USADAS INTERNAMENTE                       *
8224 REM * -----------------------------                       *
8225 REM * N$ = VARIABLE AUXILIAR                              *
8226 REM * LO = LONGITUD DE N$ EN CARACTERES                   *
8227 REM * CC = CONTADOR DE BUCLE Y PUNTERO AUXILIAR           *
8228 REM * CD = CONTADOR DE BUCLE                              *
8229 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS              *
8230 REM * NU = VARIABLE AUXILIAR                              *
8231 REM *                                                     *
8232 REM ***************************************************
8233 REM
8234 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8235 N$=P$
8236 LET TT=0
8237 GOSUB 8251
8238 N1=NU
8239 N$=Q$
8240 GOSUB 8251
8241 N2=NU
8242 IF SW=0 THEN LET TT=N1-N2:GOTO 8249
8243 IF SW=1 THEN LET TT=N1-N2:GOTO 8249
8244 IF SW=2 THEN LET TT=N1*N2:GOTO 8249
8245 IF SW=3 THEN LET TT=INT(N1/N2):GOTO 8249
8246 IF SW=4 THEN LET TT=INT(SQR(N1):GOTO 8249
8247 IF SW=5 THEN FOR CC=1 TO N1:TT=TT*CC:NEXT CC:GOTO 8249
8248 IF SW=6 THEN LET TT=N1^N2:GOTO 8249
8249 GOSUB 8263
8250 RETURN
8251 REM
8252 REM *** CAMBIO DE BASE B1 A BASE 10 ***
8253 REM
8254 LET LO=LEN(N$)
8255 LET NU=0
8256 FOR CC=1 TO LO
8257    FOR CD=1 TO B1
8258       IF MID$(S$,CD,1)<>MID$(N$,CC,1) THEN GOTO 8260
8259       LET NU=NU+INT((CD-1)*(B1^(LO-CC))+.5):LET CD=B1
8260    NEXT CD
8261 NEXT CC
8262 RETURN
8263 REM
8264 REM *** PASO DE BASE 10 A BASE B1 ***
8265 REM
8266 LET T$=""
```

```
8267 LET CC=INT(B1*((TT/B1)-INT(TT/B1))+1.5)
8268 LET T$=MID$(S$,CC,1)+T$
8269 LET TT=INT(TT/B1)
8270 IF TT>0 THEN GOTO 8267
8271 RETURN
```

EJERCICIO N.º 2

¿Cómo se formarían los números decimales en otras bases?

SOLUCION

Los números decimales en otras bases distintas de la base 10 se realizan de la misma manera. La única diferencia es que los ceros a la derecha después de la coma no se pueden suprimir. Por ejemplo, el número 12.8 en base dos (2) sería:

1100.1000

si le quitásemos los tres ceros que están más a la derecha del último uno (1) nos quedaría:

1100.1

y este número no sería 12.8, sino 12.1.

EL N.º EN BASE 3
12022.20100

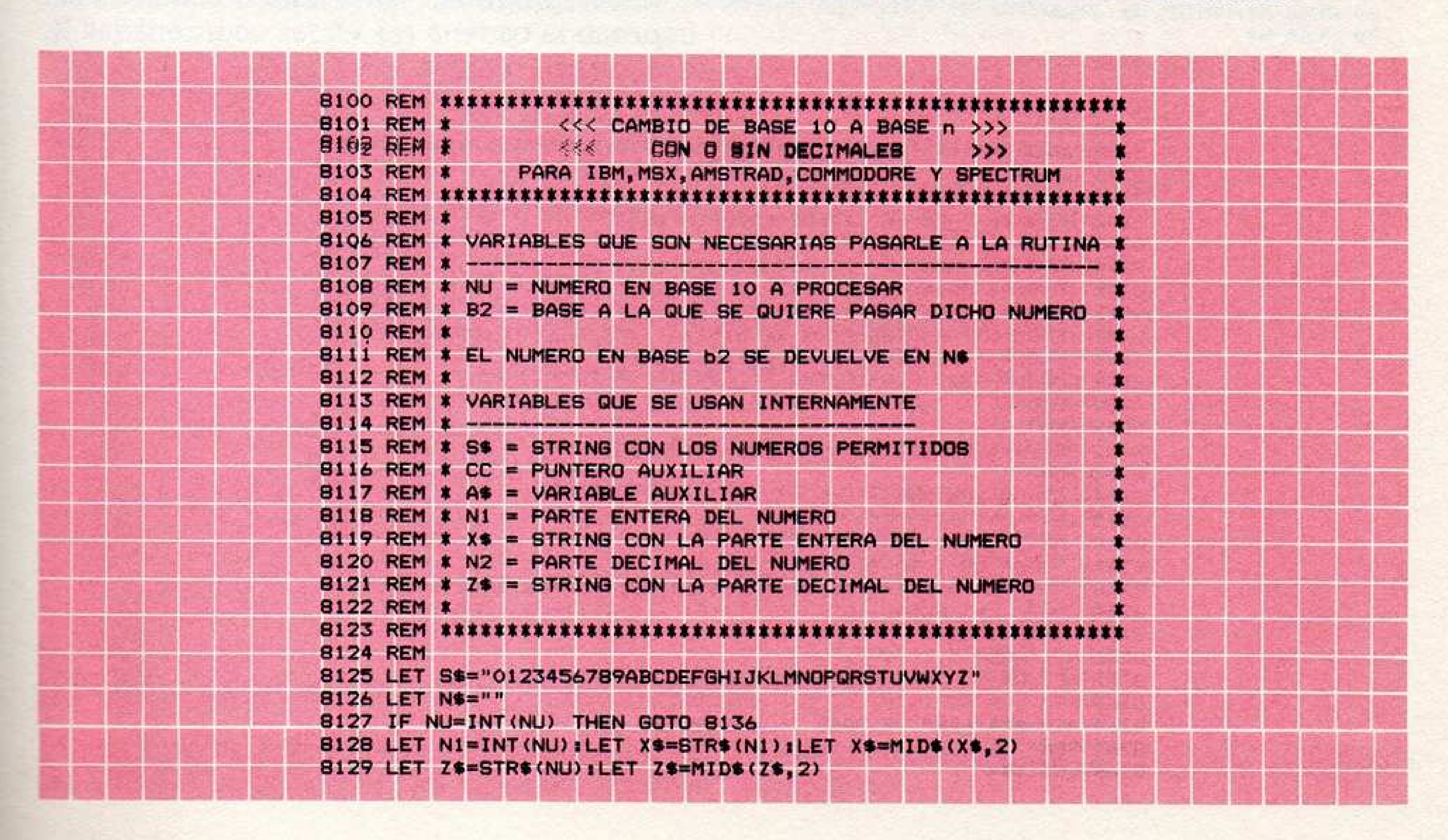
ES IGUAL AL
173.171

EN BASE 10

Fig. 4.—Cuando cambiamos de base, los ceros de la derecha de la coma sí son importantes.

Para realizar un programa de cambios de base con números decimales habrían que separar la parte entera del número de su parte decimal. Convertir la parte decimal en entera. Pasar los dos números a la base deseada. Unir la parte entera con la decimal y poner entre medias un punto.

El programa 2.6 pasa números en base 10 a otra base B1 con o sin decimales.



```
8100 REM ***********************************************
8101 REM *          <<< CAMBIO DE BASE 10 A BASE n >>>  *
8102 REM *          <<<     CON O SIN DECIMALES    >>>  *
8103 REM *      PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM *
8104 REM ***********************************************
8105 REM *                                              *
8106 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
8107 REM * -------------------------------------------- *
8108 REM * NU = NUMERO EN BASE 10 A PROCESAR            *
8109 REM * B2 = BASE A LA QUE SE QUIERE PASAR DICHO NUMERO *
8110 REM *                                              *
8111 REM * EL NUMERO EN BASE b2 SE DEVUELVE EN N$       *
8112 REM *                                              *
8113 REM * VARIABLES QUE SE USAN INTERNAMENTE           *
8114 REM * ----------------------------------           *
8115 REM * S$ = STRING CON LOS NUMEROS PERMITIDOS       *
8116 REM * CC = PUNTERO AUXILIAR                        *
8117 REM * A$ = VARIABLE AUXILIAR                       *
8118 REM * N1 = PARTE ENTERA DEL NUMERO                 *
8119 REM * X$ = STRING CON LA PARTE ENTERA DEL NUMERO   *
8120 REM * N2 = PARTE DECIMAL DEL NUMERO                *
8121 REM * Z$ = STRING CON LA PARTE DECIMAL DEL NUMERO  *
8122 REM *                                              *
8123 REM ***********************************************
8124 REM
8125 LET S$="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
8126 LET N$=""
8127 IF NU=INT(NU) THEN GOTO 8136
8128 LET N1=INT(NU):LET X$=STR$(N1):LET X$=MID$(X$,2)
8129 LET Z$=STR$(NU):LET Z$=MID$(Z$,2)
```

```
8130 LET Z$=MID$(Z$,LEN(X$)+2)
8131 LET NU=VAL(X$):GOSUB 8136
8132 LET A$=N$:LET N$=""
8133 LET NU=VAL(Z$):GOSUB 8136
8134 LET N$=A$+"."+N$
8135 RETURN
8136 LET CC=INT(B2*((NU/B2)-INT(NU/B2))+1.5)
8137 LET N$=MID$(S$,CC,1)+N$
8138 LET NU=INT(NU/B2)
8139 IF NU>0 THEN GOTO 8136
8140 RETURN
```

EJERCICIO N.º 3.

Estas rutinas son buenas para ahorrar memoria. ¿Te imaginas alguna manera de hacerlo?

SOLUCION

Las soluciones a esta cuestión son infinitas, pero el programa 2.7 nos muestra una de ellas.

Este programa ahorra memoria porque todos los números que se encuentran en las DATAS están en hexadecimal (base 16) y ocupan menos espacio.

Los programas que utilizan rutinas en código máquina y que las tienen almacenadas en líneas de DATA suelen utilizar este método.

```
10 REM ************************************
20 REM * TELEFONOS ALMACENADOS EN BASE 16 *
30 REM ************************************
40 REM
50 FOR I=1 TO 4
60 READ N$:PRINT "EL TELEFONO DE ";N$;" ES: ";
70 READ N$
80 B1=16
90 GOSUB 8900
100 PRINT NU
110 PRINT
120 NEXT I
130 END
140 REM
150 DATA "MARIA","64EA0"
160 DATA "LUIS","5E1020"
170 DATA "PEDRO (BARCELONA) ","37E52CC0"
180 DATA "ESTER","70D0A3"
```

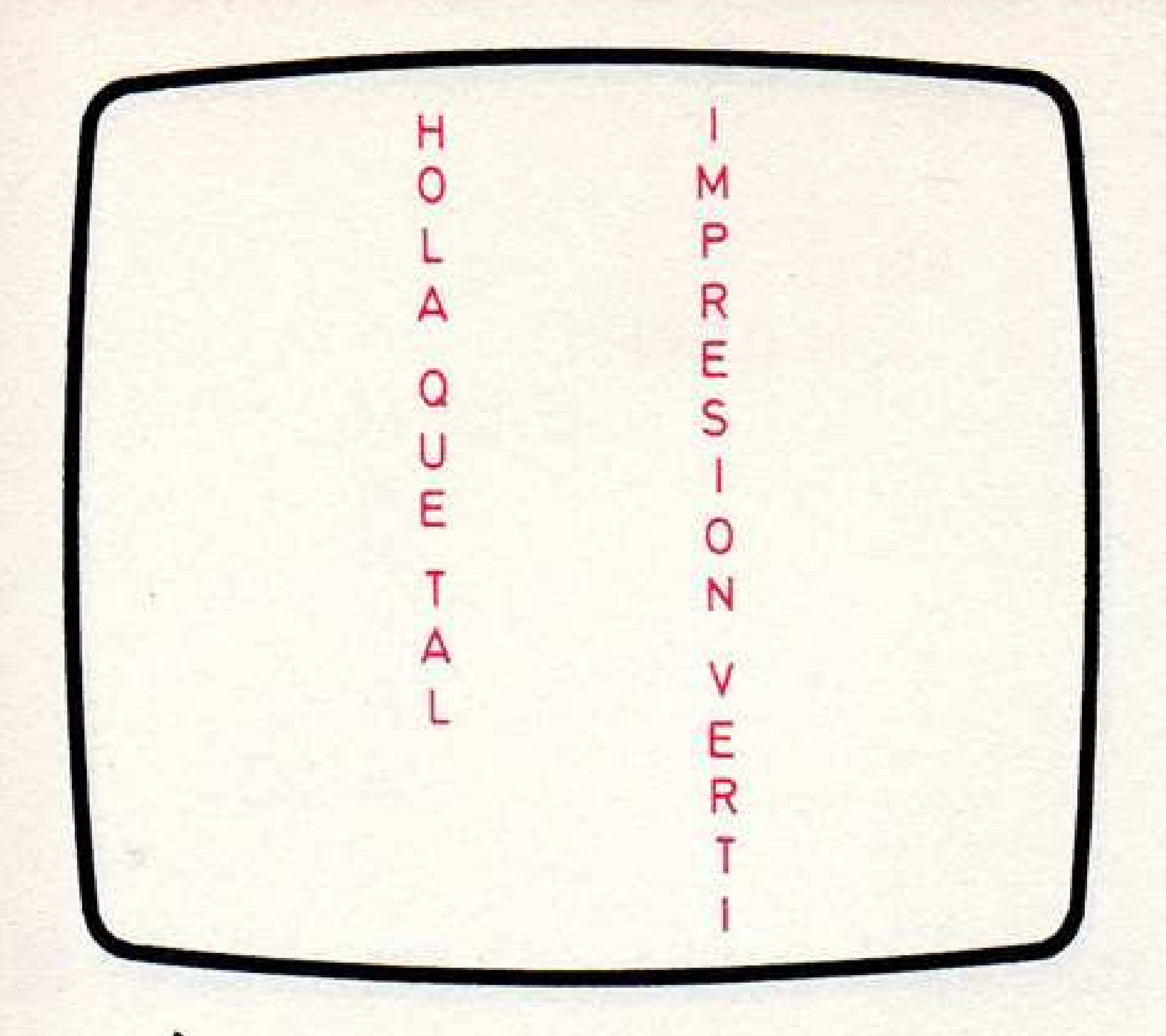
Impresión de mensajes en distintos sentidos

Como hemos visto anteriormente, la presentación de mensajes en la pantalla de nuestro ordenador es una de las cosas que más tenemos que cuidar a la hora de hacer nuestros programas.

Una vez que hemos aprendido cómo imprimir mensajes de formas distintas aprenderemos cómo imprimir mensajes en distintos sentidos y orientaciones.

El programa que viene a continuación imprime la cadena M$ en las coordenadas X, Y de arriba a abajo.

```
1000 REM ***************************************************
1010 REM *        <<< IMPRESION VERTICAL TELETIPO >>>       *
1020 REM *                                                  *
1030 REM *      PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM   *
1040 REM ***************************************************
1050 REM *                                                  *
1060 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
1070 REM * ------------------------------------------------ *
1080 REM * M$ = MENSAJE A IMPRIMIR                          *
1090 REM * X  = COLUMNA DONDE SE IMPRIME                    *
1100 REM * Y  = FILA DONDE SE EMPIEZA A IMPRIMIR            *
1102 REM * TI = RETARDO ENTRE LETRA Y LETRA                 *
1105 REM *                                                  *
1110 REM * LA RUTINA NO DEVUELVE VALORES                    *
1120 REM *                                                  *
1130 REM * VARIABLES QUE SE USAN INTERNAMENTE               *
1140 REM * ----------------------------------               *
1150 REM * CC = CONTADOR DE BUCLE                           *
1160 REM * CD = CONTADOR DE BUCLE                           *
1170 REM ***************************************************
1180 REM
1190 FOR CC=1 TO LEN(M$)
1200    LOCATE Y-1+CC,X
1210    PRINT MID$(M$,CC,1)
1220    FOR CD=1 TO TI
1230    NEXT CD
1240 NEXT CC
1250 RETURN
```



El único truco del programa es que en vez de ir variando el valor de la columna X, como veíamos en el fascículo anterior, se varía el de la fila Y. Según esto, todos los programas de impresión de mensajes que hemos visto anteriormente se pueden arreglar para que funcionen imprimiendo mensajes en vertical en vez de en horizontal.

Como ejemplo aparece el programa 2.9 para que veas cómo se pueden transformar. Los demás te los dejamos a ti para que vayas cogiendo práctica.

Fig. 5.—Con el programa 2.8 podemos escribir mensajes de arriba a abajo.

```
8000 REM *************************************************************
8001 REM *     <<< IMPRESION VERTICAL CON DESPLAZAMIENTO >>>          *
8002 REM *                                                            *
8003 REM *   PARA MSX,IBM,AMSTRAD,COMMODORE Y SPECTRUM                *
8004 REM *************************************************************
8005 REM *                                                            *
8006 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA                  *
8007 REM * ----------------------------------------                   *
8008 REM *                                                            *
8009 REM * X  = COLUMNA DONDE EMPIEZA LA IMPRESION                    *
8010 REM * Y  = FILA DONDE COMIENZA LA IMPRESION                      *
8011 REM *                                                            *
8012 REM * M$ = MENSAJE A IMPRIMIR                                    *
8013 REM *                                                            *
8014 REM * LF = ALTURA MAXIMA DEL MENSAJE                             *
8015 REM * TI = PAUSA ENTRE LETRAS                                    *
8016 REM *                                                            *
8017 REM * LA RUTINA NO DEVUELVE VALORES                              *
8018 REM *                                                            *
8019 REM * VARIABLES USADAS INTERNAMENTE POR LA RUTINA                *
8020 REM * -------------------------------------------                *
8021 REM *                                                            *
8022 REM * CC = CONTADOR DE BUCLE                                     *
8023 REM * CD = CONTADOR DE BUCLE                                     *
8024 REM * CE = CONTADOR DE BUCLE                                     *
8025 REM * B$ = STRING DE BLANCOS AUXILIAR                            *
8026 REM *                                                            *
8027 REM *************************************************************
8028 REM
8029 LET B$=""
8030 FOR CC=1 TO LF+2
8031     LET B$=B$+" "
8032 NEXT CC
8033 FOR CC=1 TO LEN(B$+M$)+1
8034     FOR CE=1 TO LF
8035         LOCATE Y-1+CE,X
8036         PRINT MID$(B$+M$+B$,CE+CC,1)
8037     NEXT CE
8038     PLAY "L64 D"
8039     FOR CD=1 TO TI
8040     NEXT CD
8041 NEXT CC
8042 RETURN
```

Nota para los usuarios del SPECTRUM

Las variables índices de los bucles, en el SPECTRUM, sólo pueden tener un carácter como nombre. Por eso, si vemos un programa que pone:

FOR CC = 1 TO 30

este nombre (CC) no nos será aceptado.

Para remediar esto se propone coger la segunda letra del nombre de la variable. Por ejemplo, si aparece:

FOR CE = 1 TO TI

nosotros pondremos:

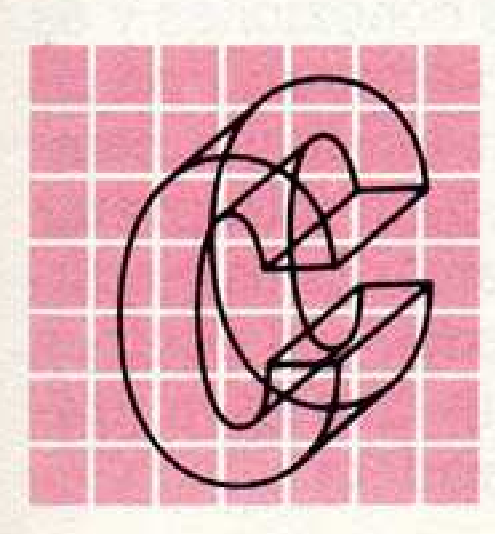
FOR E - 1 TO T1

Hay que tener cuidado con que esta variable (E) no exista ya en el programa.



Fig. 6.—¿Sabrías hacer un programa que imprimiese mensajes de esta manera?

Tarjeta con dispositivos de interfaz



ON la tarjeta de ampliación de puertos, que ya hemos definido para poder ampliar las posibilidades de nuestro ordenador personal, se pueden realizar conexiones directas con algunos equipos periféricos que posean las mismas características eléctricas que los circuitos utilizados.

Para el puerto de salida se ha empleado un circuito de biestables de tipo D 74LS374, con posibilidad de control de la salida para ponerla en tercer estado. De momento la patilla de control la conectaremos a masa, a través de un puente, con lo que la salida estará activada en todo momento. Al conectarla mediante un puente podremos desactivarla manualmente o bien conectarla a la salida de otro de los puertos para controlarla mediante programa. La salida de estos circuitos es de tipo amplificador TTL, pudiendo dar una corriente limitada. En estado 1, puede suministrar unos 2,5 miliamperios, en cuyo caso la tensión de salida desciende hasta unos 3 voltios. En estado 0, admite una corriente de unos 24 miliamperios. Estas cifras nos indican que será necesario utilizar algún circuito de amplificación para poder suministrar la corriente suficiente para activar la mayoría de los dispositivos prácticos.

El puerto de entrada, realizado con el circuito 74LS244, presenta las características propias de una entrada TTL-LS normal, es decir, considera como nivel 1 los valores de tensión por encima de 2,4 voltios, suministrando en ese caso 40 microamperios, y como nivel 0 a valores de entrada por debajo de 0,8 voltios. Los valores intermedios comprendidos entre 0,8 y 2,4 voltios se consideran no válidos, por reducir el margen de ruido. Sin embargo, este circuito presenta la ventaja de incluir en su característica de entrada una cierta histéresis, que le asegura gran inmunidad al ruido. La histéresis es el comportamiento que presentan muchos fenómenos físicos, de seguir una trayectoria de evolución diferente para valores crecientes que para valores decrecientes de la magnitud que los activa. Puede representarse mediante el diagrama de la figura.

Para valores crecientes de la señal de entrada es necesario superar el valor indicado por V2, para que la salida pase al estado 1. Por el contrario, para que la salida vuelva al estado 0, es necesario que la tensión de entrada alcance el valor V1. Variaciones compren-

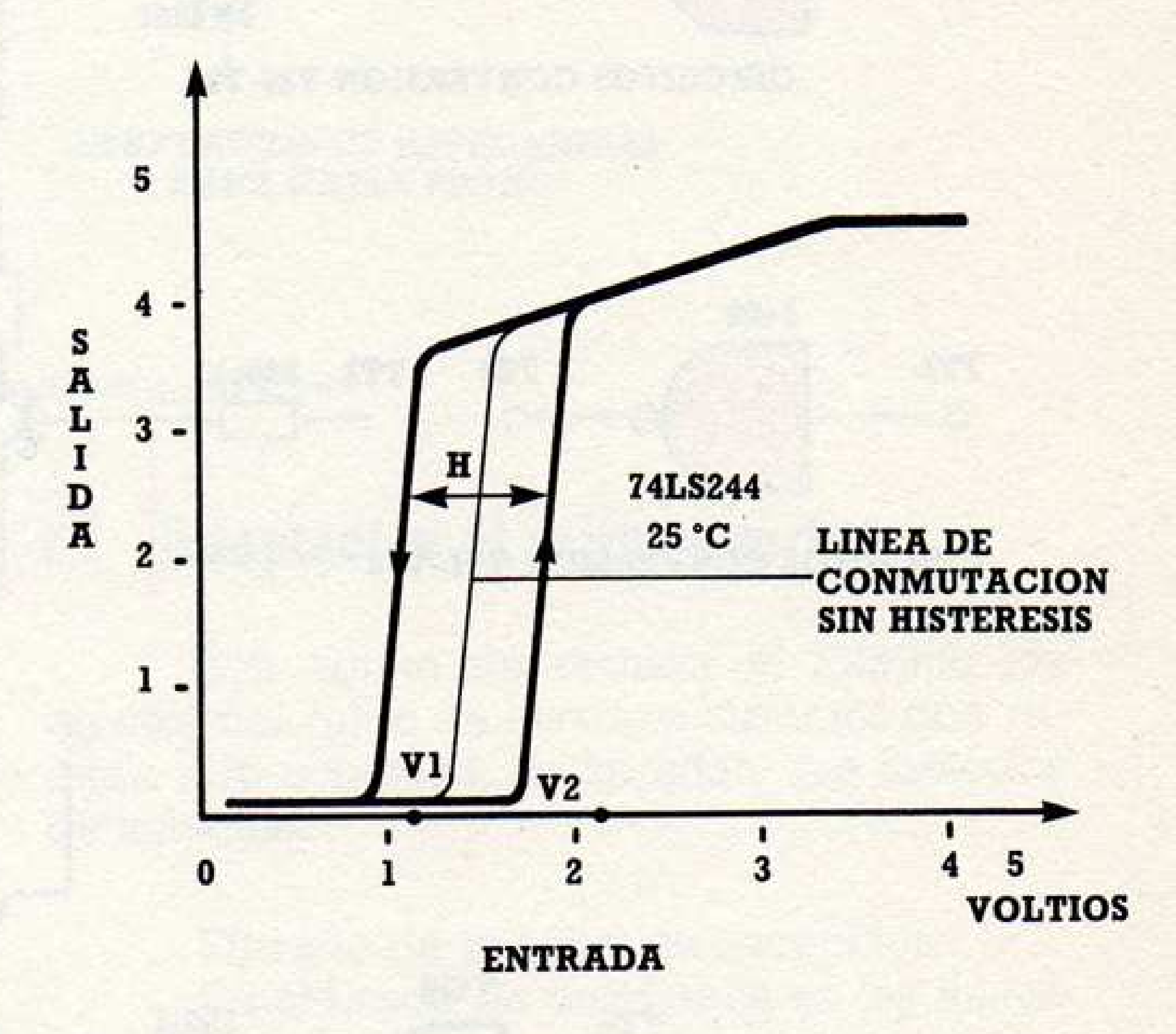


Fig. Diagrama de histéresis del circuito de entrada de 74LS244.

didas entre V1 y V2 no originan cambio de estado en la salida. Es decir, un ruido superpuesto a la señal realmente transmitida por el amplificador del circuito origen no se transformará en nivel válido en el circuito de destino mientras no sobrepase una cierta cantidad sobre el valor del nivel de conmutación. Esta separación aumenta en nivel de ruido en el valor de la histéresis H. Esta característica es altamente conveniente en todos los circuitos que se conecten a otros sistemas y por ello es empleada en todas las interfaces normalizadas de conexiones entre equipos.

Con este tipo de niveles eléctricos podemos conectar equipos próximos y de iguales características. Sin embargo, lo normal será que los equipos que deseamos conectar posean otras características o se encuentren alejados por alguna distancia y sea necesario utilizar cables largos. Para estos casos será necesario proteger las conexiones según varios puntos de vista. Veamos las diferentes situaciones y las soluciones más convenientes a adoptar.

Adaptación de niveles

Las salidas y entradas de la tarjeta de ampliación de puertos son de tipo TTL-LS. Las posibles conexiones normalizadas más comunes son:

TTL

RS-232 o V28

Bucle de corriente

En la figura se muestran diferentes circuitos con la indicación de los niveles a los que convierten. El empleo de circuitos integrados específicos para cada aplicación, simplifica grandemente el diseño de las conexiones y es la solución recomendable. Sin embargo, es interesante conocer la solución mediante componentes discretos, pues ayuda a interpretar la complejidad real del problema. Estos circuitos los utilizaremos en la conexión con equipos preparados para interfaz serie como son: trazador, ratón y tableta digitalizadora.

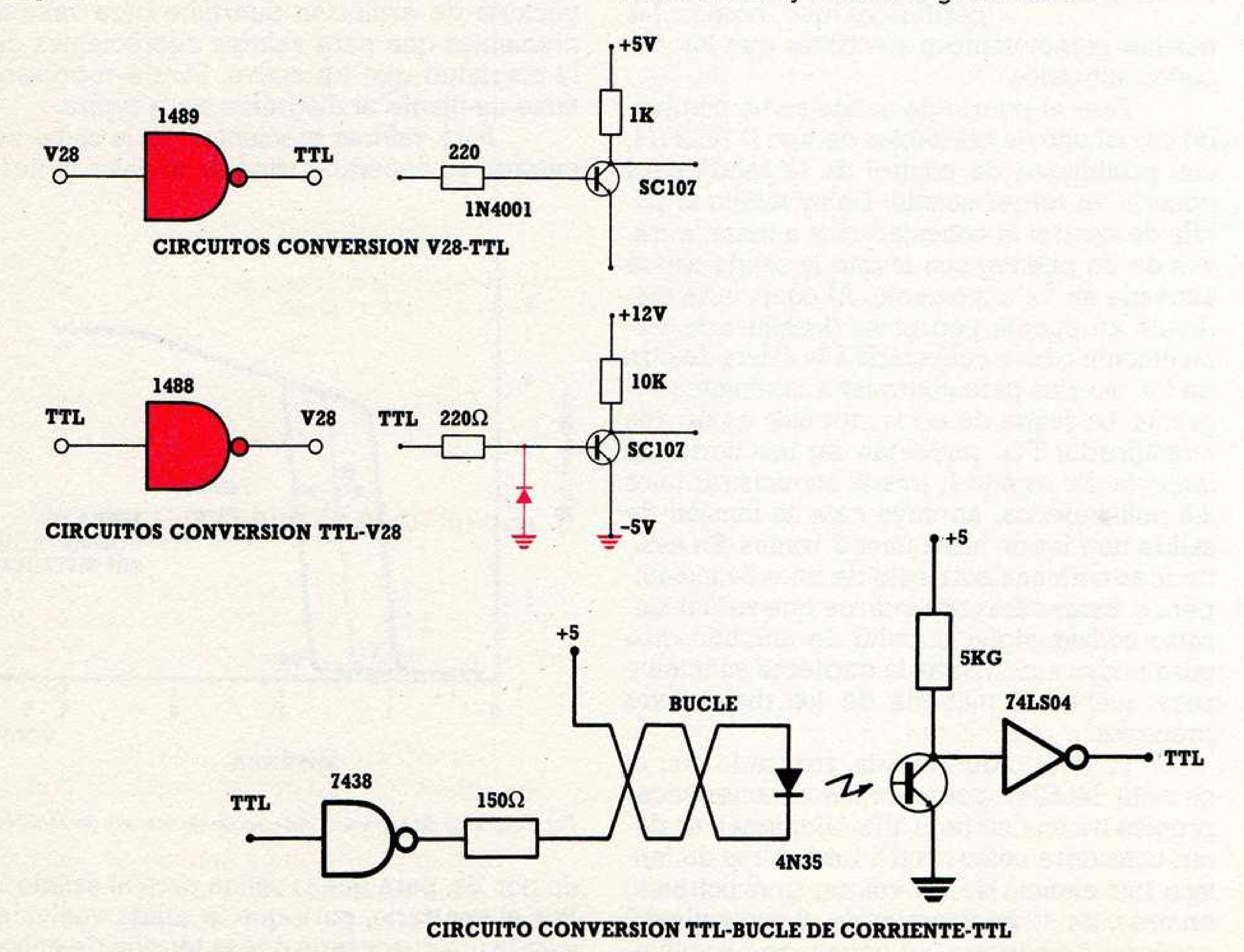


Fig. Circuitos para adaptación de niveles.

Adaptación de impedancias a la línea de conexión

Los cables utilizados en las conexiones no se comportan como si fueran un cortocircuito, pues presentan cierta impedancia al paso de la corriente. Cuando la conexión es de cierta longitud, es necesario considerar los efectos de esta impedancia y también los efectos producidos por las diferencias entre ellos, porque no todos los elementos conectados poseen la misma impedancia característica. El principal efecto es el de los "rebotes" o reflexiones, que originan falsos pulsos en el circuito receptor. En la figura se muestra una interpretación gráfica del fenómeno, sobre el circuito que se indica.

El problema es todavía más complejo si se analiza con detalle. La impedancia de entrada de un circuito TTL-LS no es constante para los valores de entrada que puedan presentarse. Esto hace que si se desea estimar con mucha exactitud el comportamiento en condiciones reales, sea necesario establecer un modelo complejo de la variación de impedancia. En la mayoría de los casos prácticos es suficiente el establecer el equilibrado de impedancias en su parte resistiva, además de conservar el nivel de continua del punto de trabajo. En la figura se muestra la conexión típica de una línea de cierta longitud de cable plano, con las resistencias de adaptación a la impedancia característica.

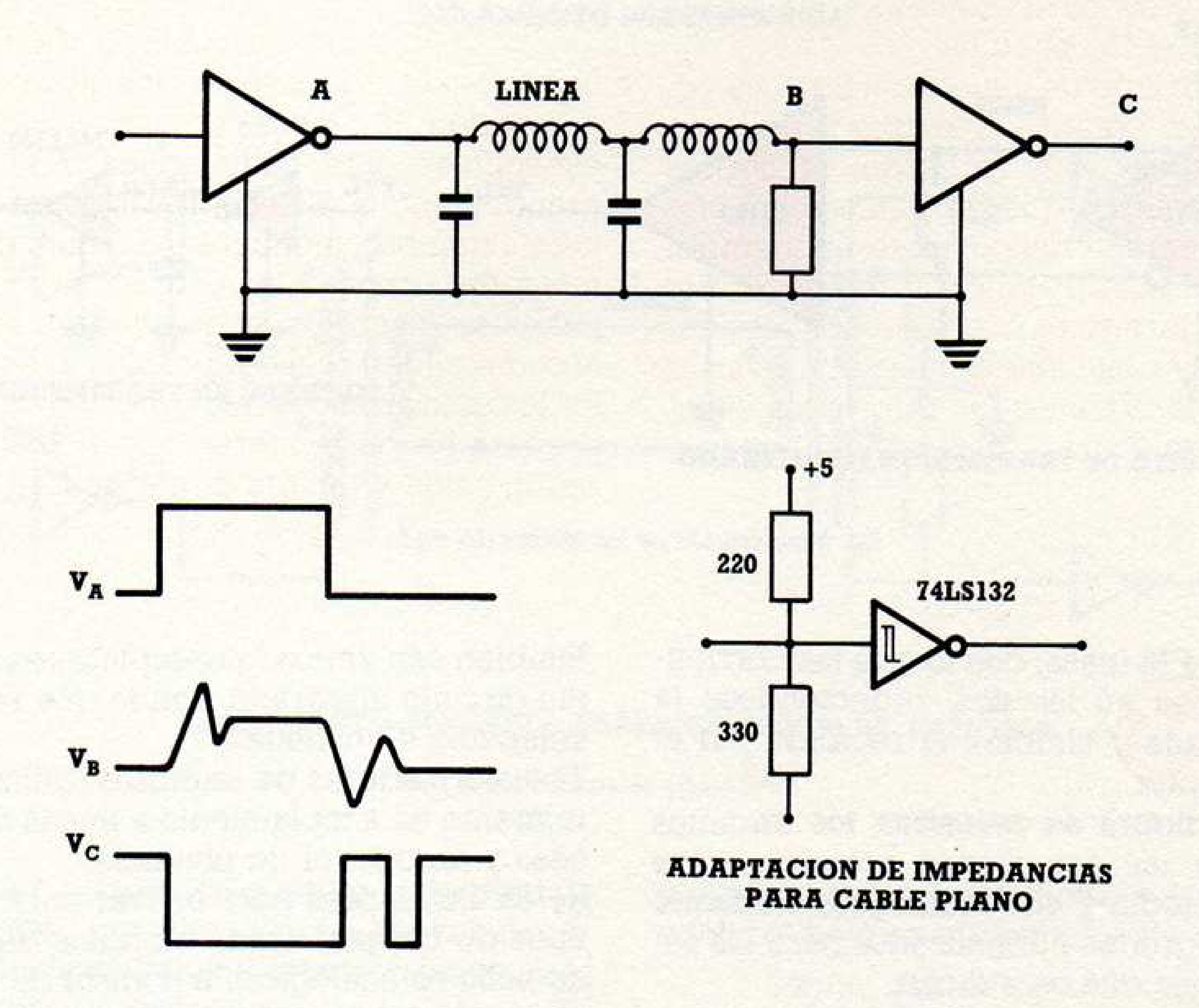


Fig. Efectos de la impedancia en las líneas.

Adaptación al ruido exterior

Con objeto de reducir al mínimo los efectos del ruido de los otros circuitos que rodean a la conexión, se adoptan dos métodos de solución:

Filtrado de la señal en recepción
Equilibrado de corrientes en las líneas

Mediante el primer método se reduce en lo posible el ancho de banda de los circuitos utilizados, empleando para ello circuitos pasivos de bobinas y condensadores o bien con filtros activos con amplificadores opera-

cionales, según los niveles de la señal. En los circuitos de conexión en paralelo de las impresoras se conecta un condensador a masa de un valor entre 100 y 1.000 picofaradios, pero siempre que el circuito receptor posea entrada con histéresis, pues de lo contrario el remedio podría ser peor que la enfermedad.

Mediante el segundo método, se establece la conexión utilizando líneas simétricas con relación a la masa, con lo que las corrientes son iguales en las dos, reduciéndose la energía radiada y también la recibida por el circuito receptor.

En la figura se muestran los circuitos utilizados en las conexiones actuales entre equipos de media y alta velocidad, mediante circuitos integrados normalizados para las recomendaciones que se indican.

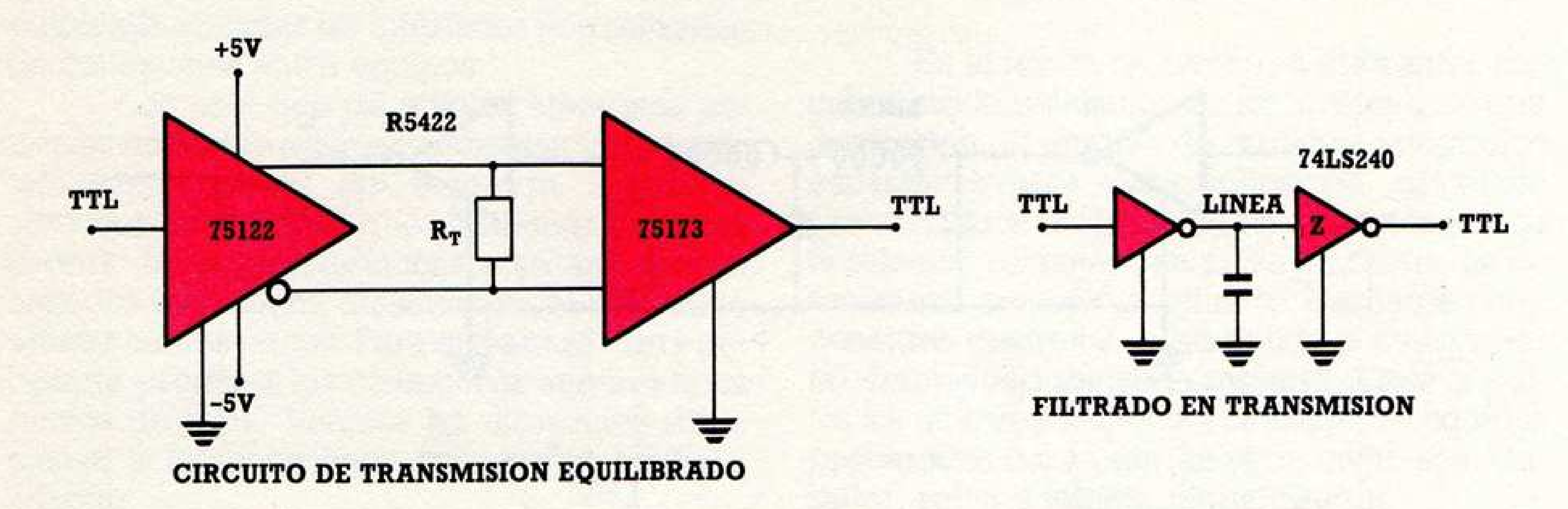


Fig. Minimización de los efectos del ruido.

Aislamiento de los circuitos

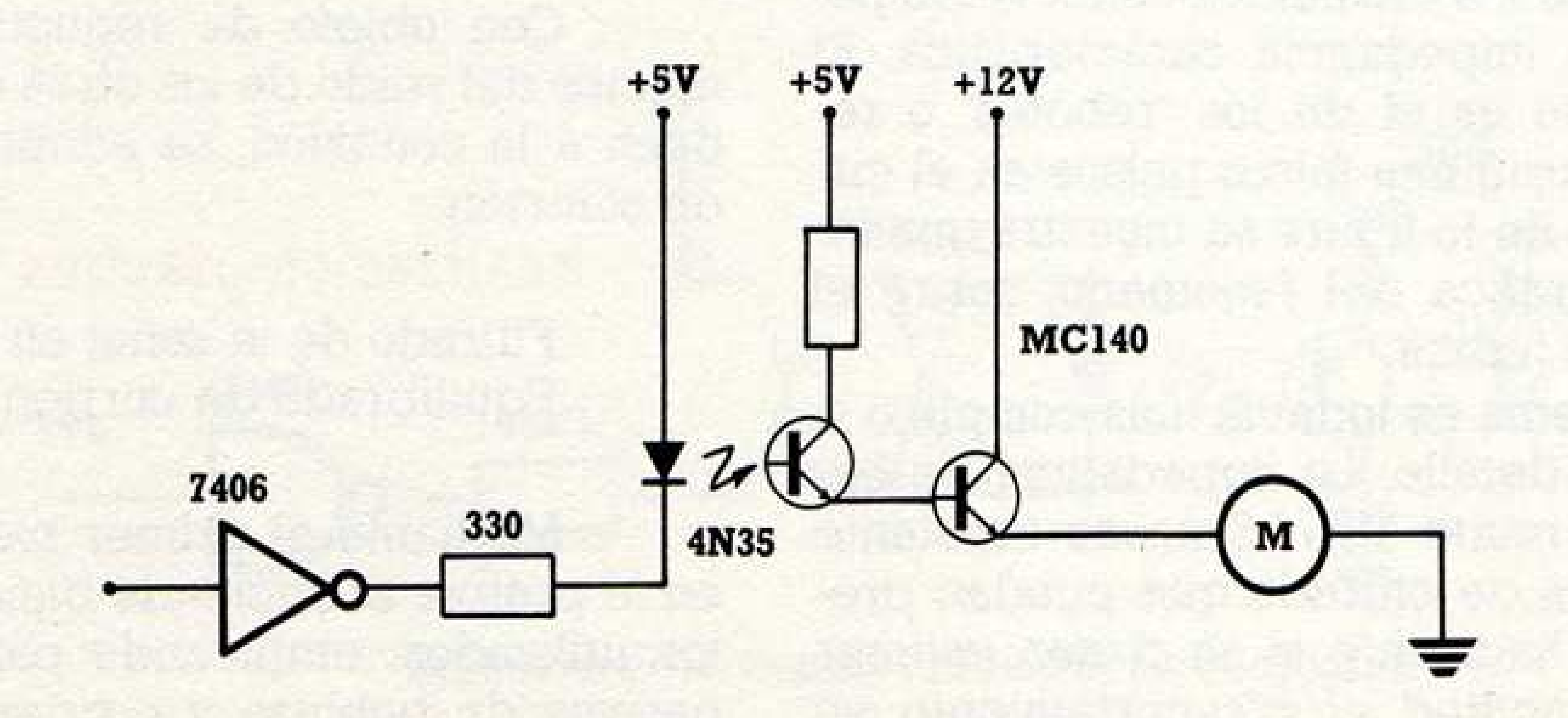
Si los sistemas que se desean conectar no poseen la misma masa de referencia o no conviene que se establezcan varios puntos de conexión, pueden utilizarse varias técnicas de aislamiento:

Fotoacopladores. Existen dos tipos básicos en el mercado: los de un diodo LED acoplado con un fototransistor y los que además incluyen otro transistor acoplado en continua, para aumentar la ganancia del conjunto. Se fabrican también con varios fotoacopladores en un mismo circuito integrado, con lo que resulta muy compacto el montaje.

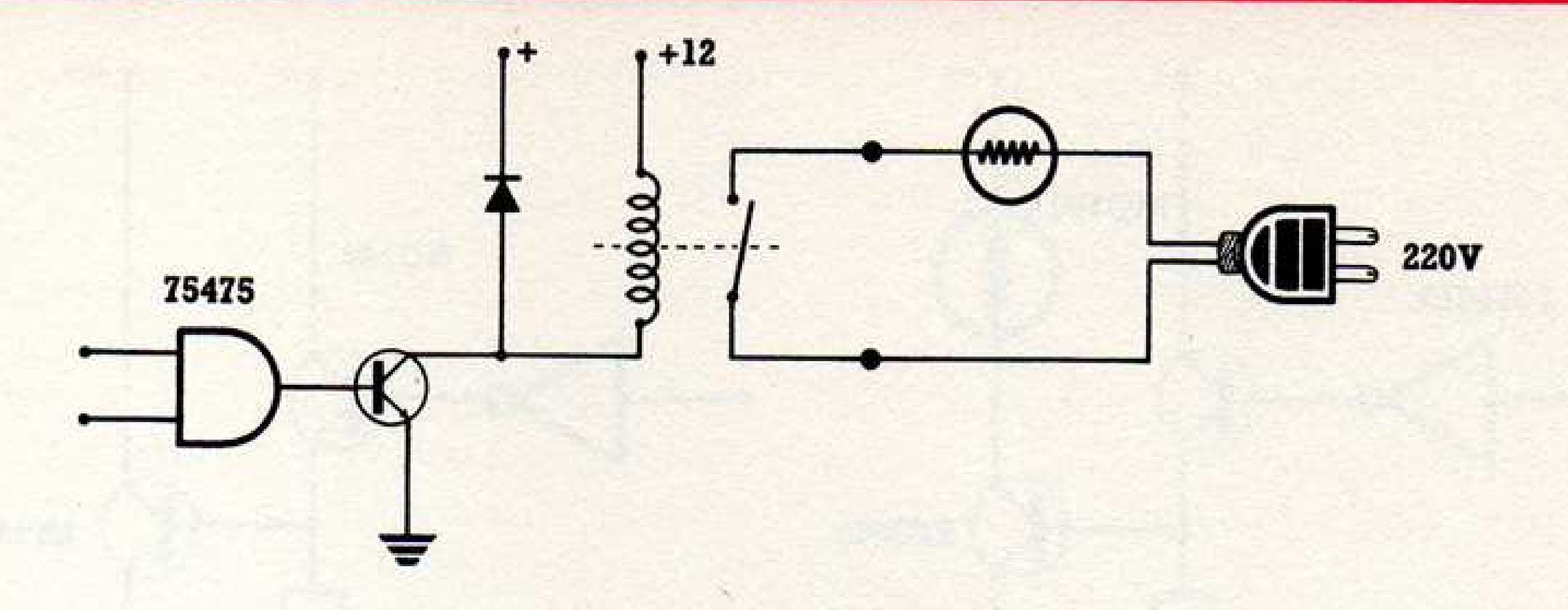
Transformadores de impulsos. Utilizado generalmente en acoplamiento a líneas de teleproceso y en control de potencia.

Relés. Se emplea para activar muy diferentes tipos de cargas, desde circuitos de muestreo de señales analógicas a control de cargas de mucha potencia, conectadas directamente a la red eléctrica.

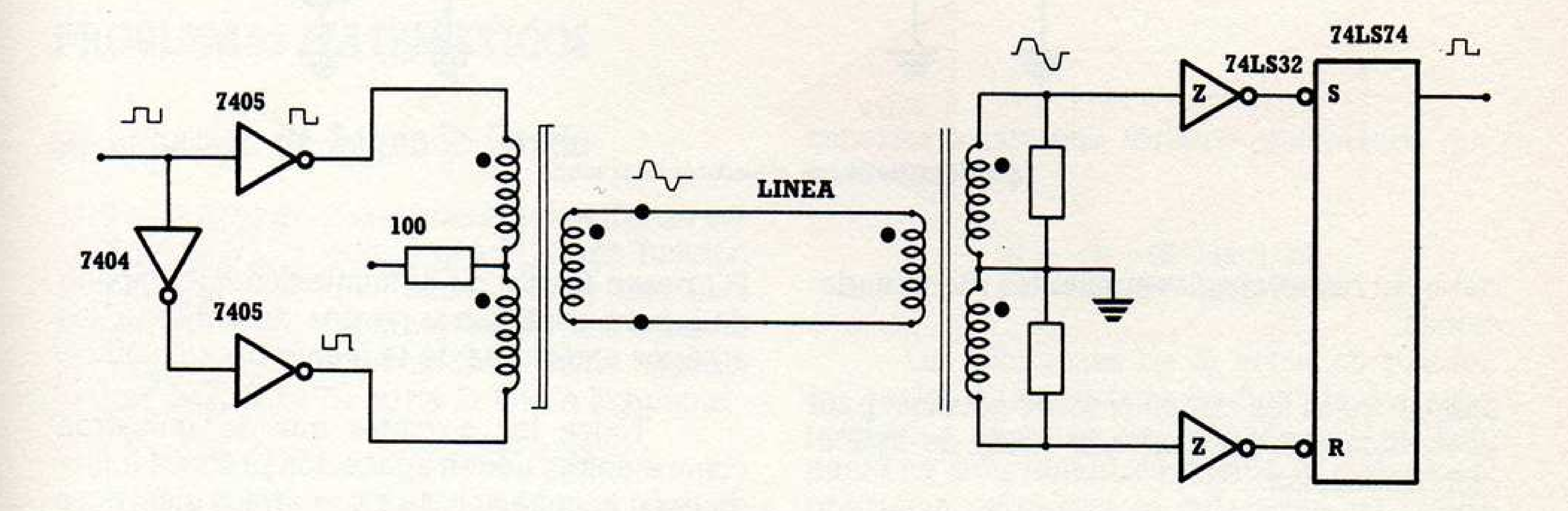
Con los circuitos de la figura pueden activarse los circuitos exteriores sin que haya posibles efectos por conexiones de masa.



AISLAMIENTO MEDIANTE FOTOACOPLADOR



AISLAMIENTO MEDIANTE RELE



AISLAMIENTO MEDIANTE TRANSFORMADOR DE IMPULSOS

Fig. Circuitos de aislamiento.

Igualmente en conexión a equipos de medida aislada, como, por ejemplo, en los relacionados con electroquímica o bioelectrónica, será necesario utilizar aislamiento en las etapas de entrada.

Alimentación de actuadores de potencia

Además de los circuitos de aislamiento, que permiten alimentar todo tipo de cargas, resulta conveniente disponer de etapas de amplificación que puedan activarse agrupando varios bits. Así, por ejemplo, pueden controlarse motores de continua, gobernando con un bit el arranque/parada y con el otro el sentido del movimiento. La circuitería necesaria depende básicamente de la potencia del motor a controlar y de la velocidad de conmutación. Los transistores se utilizan como conmutadores si/no. Circuitos de este tipo los emplearemos en los proyectos relacionados con equipos móviles y maquetas de robot.

Según que el circuito de carga pueda ir flotante respecto a la masa o conectado a ella, se utilizará un amplificador de potencia de uno de los tipos indicado.

En la práctica, todos los circuitos de conexión incluyen componentes para tener en cuenta los criterios indicados, para optimizar en lo posible la comunicación al mínimo coste.

Realización práctica

Con los circuitos indicados para cada uno de los casos podemos realizar una tarjeta que incluya un número de circuitos necesarios para una aplicación práctica. Es importante considerar que tanto los circuitos del bus del ordenador como los de la tarjeta de ampliación de puertos son muy delicados, por lo que

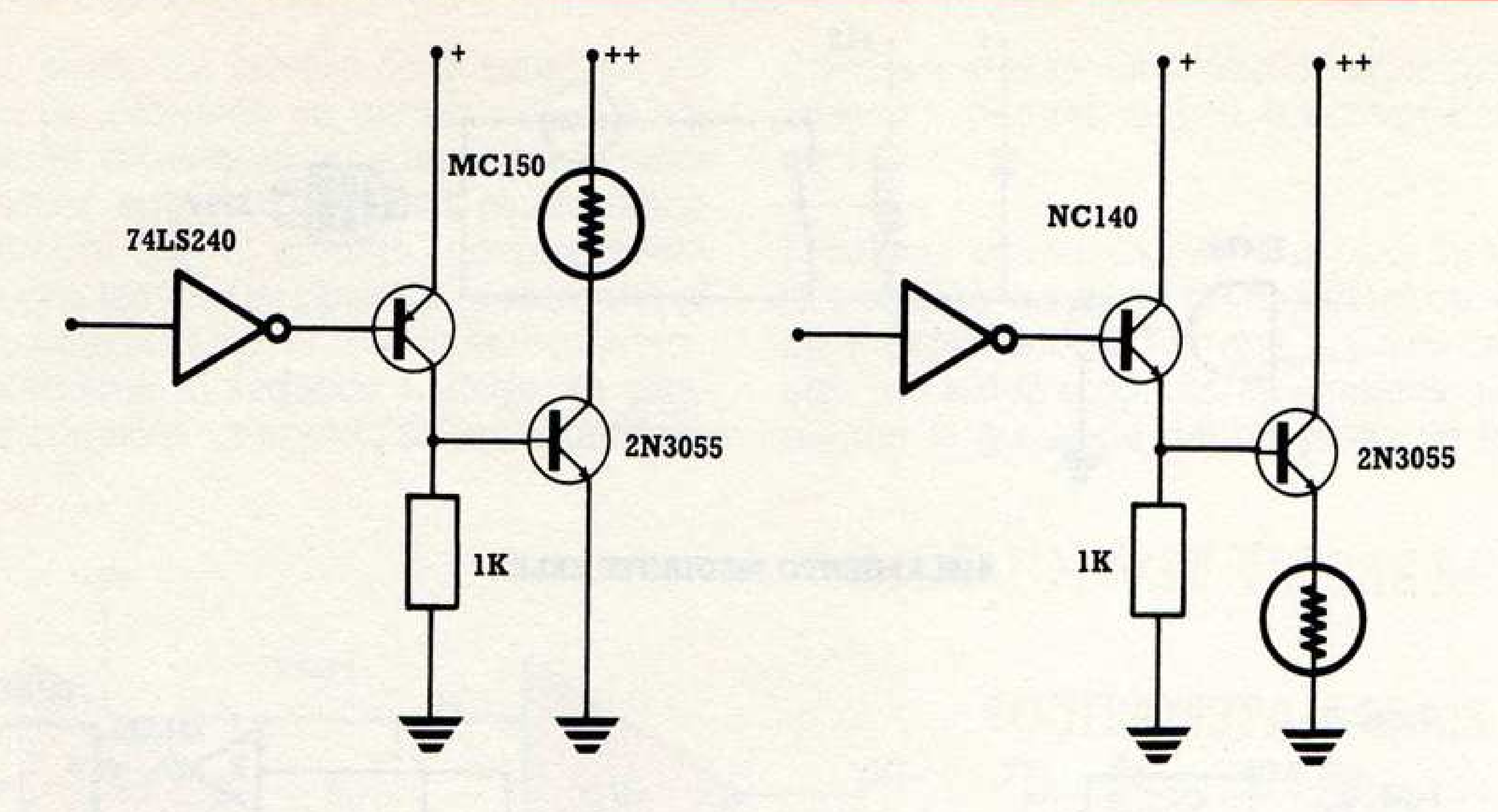


Fig. Alimentación de actuadores de potencia.

deberán respetarse las siguientes recomendaciones:

Montar todos los circuitos sobre zócalos, para la verificación del cableado antes de aplicación de tensión.

Conectar las tarjetas con la máquina apagada.
Utilizar un soldador de poca potencia, con tiempo de soldadura inferior a los 10 segundos, si se suelda algún componente directamente.

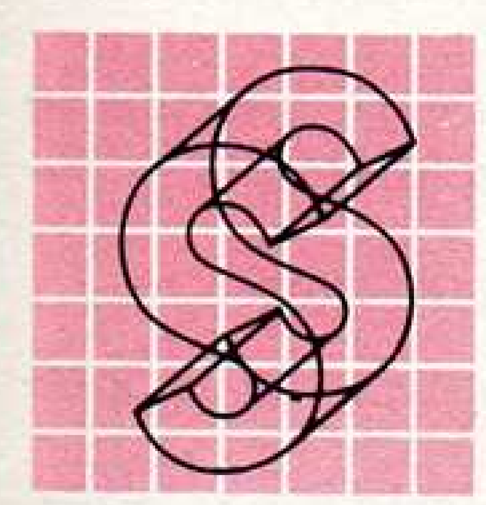
Colocar alguna ranura de clave para evitar la inversión del conector.
Utilizar alimentación independiente de la del ordenador si el consumo es superior a los valores especificados. En general, si se sobrepasan los 200 miliamperios es conveniente la alimentación independiente. Solamente el IBM-PC posee fuente de alimentación con capacidad para suministrar varios amperios a los equipos exteriores de la fuente de + 5 voltios.

Todos los circuitos que se muestran como ejemplo tienen aplicación práctica inmediata en la conexión de los aparatos eléctricos que nos rodean y que pueden ser controlados por el ordenador. Una vez que tengamos familiaridad con las conexiones sencillas estaremos en disposición de emprender proyectos más complejos que requieran dispositivos más sofisticados. También deberemos considerar la programación para ejecutar las aplicaciones asociadas con los equipos conectados, pues hasta ahora solamente hemos contemplado los casos de circuitos que se activan con un bit o que suministran a la entrada una información binaria.

NATURALEZA Y TECNOLOGIA

PROBLEMAS MATEMATICOS

Ecuación de Segundo Grado



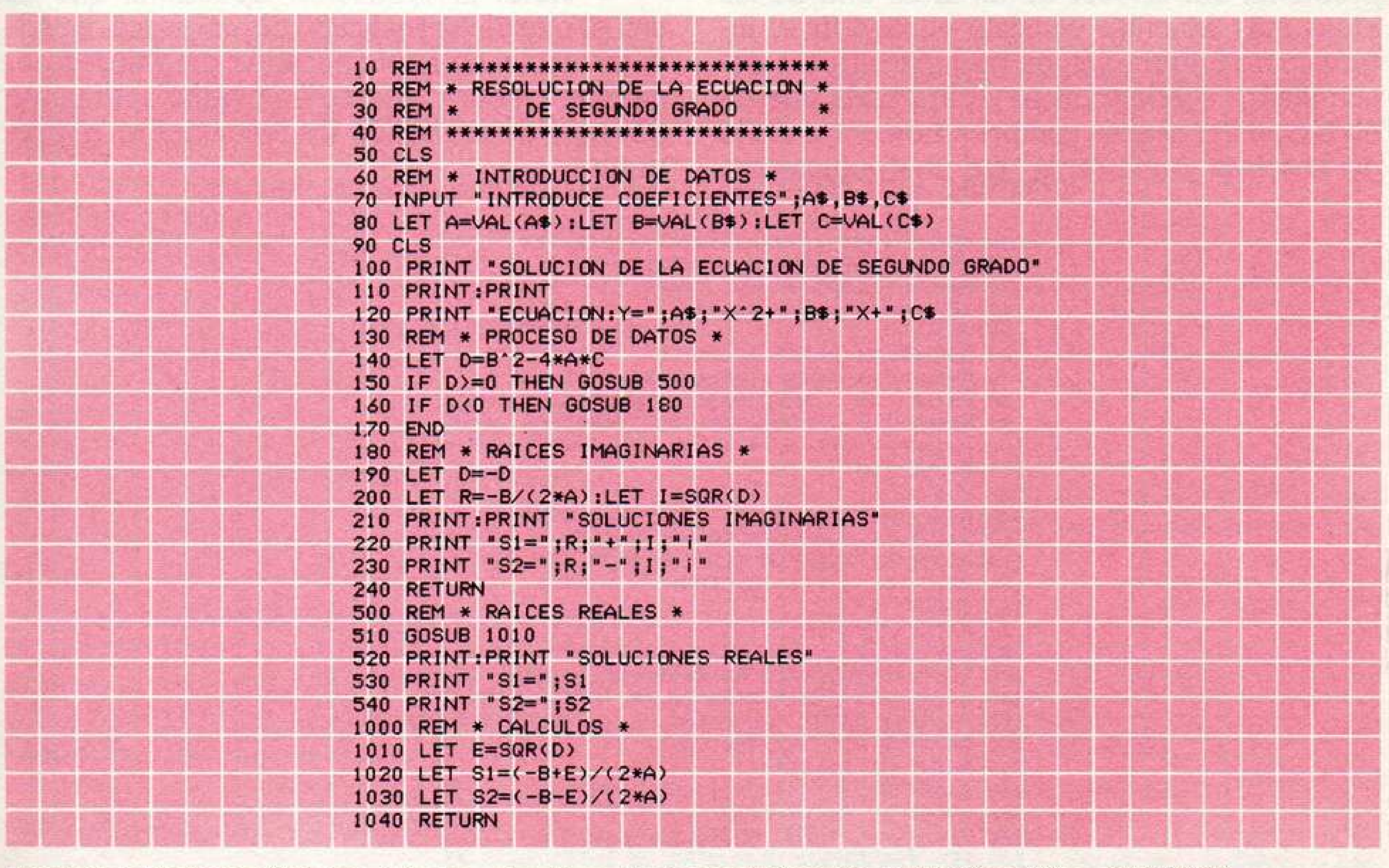
E denomina ecuación de segundo grado a una función que depende de una sola variable que llega a tomar el exponente 2 como mucho. Tendría la forma siguiente:

$$y = ax^2 + bx + c$$

La ecuación de segundo grado tiene dos soluciones. Para determinarlas existe una fórmula matemática que es la siguiente:

$$s1 = - b + (b^2-4ac)/2a$$
$$s2 = - b -(b^2-4ac)/2a$$

Las soluciones de la ecuación pueden ser reales o imaginarias en función del valor positivo o negativo del término b^2-4ac. Este punto ha sido tenido en cuenta al elaborar el programa dado que el ordenador no puede calcular raíces de números negativos.



```
10 REM ******************************
20 REM * RESOLUCION DE LA ECUACION *
30 REM *      DE SEGUNDO GRADO      *
40 REM ******************************
50 CLS
60 REM * INTRODUCCION DE DATOS *
70 INPUT "INTRODUCE COEFICIENTES";A$,B$,C$
80 LET A=VAL(A$):LET B=VAL(B$):LET C=VAL(C$)
90 CLS
100 PRINT "SOLUCION DE LA ECUACION DE SEGUNDO GRADO"
110 PRINT:PRINT
120 PRINT "ECUACION:Y=";A$;"X^2+";B$;"X+";C$
130 REM * PROCESO DE DATOS *
140 LET D=B^2-4*A*C
150 IF D>=0 THEN GOSUB 500
160 IF D<0 THEN GOSUB 180
170 END
180 REM * RAICES IMAGINARIAS *
190 LET D=-D
200 LET R=-B/(2*A):LET I=SQR(D)
210 PRINT:PRINT "SOLUCIONES IMAGINARIAS"
220 PRINT "S1=";R;"+";I;"i"
230 PRINT "S2=";R;"-";I;"i"
240 RETURN
500 REM * RAICES REALES *
510 GOSUB 1010
520 PRINT:PRINT "SOLUCIONES REALES"
530 PRINT "S1=";S1
540 PRINT "S2=";S2
1000 REM * CALCULOS *
1010 LET E=SQR(D)
1020 LET S1=(-B+E)/(2*A)
1030 LET S2=(-B-E)/(2*A)
1040 RETURN
```

Programa 1: Programa válido para todos los ordenadores. En el caso del Spectrum sustituir línea 170 por GOTO 9999.

```
SOLUCION DE LA ECUACION DE SEGUNDO GRADO

ECUACION:Y=1X^2+10X+9

SOLUCIONES REALES
S1=-1
S2=-9
OK
```

Fig. 1.

Cálculo de integrales definidas

El concepto de integral es uno de los más importantes de las matemáticas. Su aplicación inmediata es el cálculo de áreas.

Para la resolución de estos problemas se utilizan las denominadas integrales definidas. Estas se diferencian de las indefinidas o primitivas en que el resultado final es un número y no una función. Las integrales definidas tienen dos extremos de integración que forman un intervalo.

Existen muchas formas de resolver el problema del cálculo de un área. Una de las ideas más sencillas consiste en dividir el intervalo de integración en pequeños trozos denominados subintervalos, y para cada uno de ellos calcular el área comprendida entre el eje de abscisas y una recta que une puntos consecutivos de diferentes intervalos. Si sumamos todas las áreas así obtenidas obtendremos una aproximación de la integral. Esta se aproximará más a la realidad cuanto mayor sea el número de puntos en el cual dividamos el intervalo de integración.

Desarrollo del programa

El programa realiza la integración numérica de una función de x cualquiera, definida por el usuario, por el método de los trapecios antes descrito.
Como datos precisamos:

- La función a integrar
- Los extremos del intervalo de integración
- El número de puntos en los que se quiere subdividir el intervalo.

Como resultados obtendremos:

- El resultado aproximado de la integral
- La gráfica de la función en el intervalo en el que estamos trabajando.

Nota: El ordenador Spectrum, que es con el que se ha realizado el programa, no admite funciones exponenciales de valores negativos. Deberá tenerse en cuenta este punto a la hora de introducir las funciones en el programa.

```
 10 REM ******************************
 20 REM *  INTEGRACION NUMERICA     ***
 30 REM * METODO DE LOS TRAPECIOS  ***
 40 REM ******************************
 50 CLS
 60 BORDER 2: PAPER 7: INK 0
 70 PRINT "*******************************"
 80 PRINT "**   INTEGRACION NUMERICA     **"
 90 PRINT "**  METODO DE LOS TRAPECIOS   **"
100 PRINT "*******************************"
110 PAUSE 100
120 CLS
130 REM ***INTRODUCCION DE LA FUNCION***
140 PRINT AT 3,0;"TECLEA LA FUNCION A INTEGRAR"
145 PRINT AT 4,0;"LA FUNCION DEBE SER CONTINUA "
150 PRINT AT 6,10;"FUNCION="
160 INPUT Y$
170 PRINT AT 6,19;Y$
180 DEF FN J(X)=VAL Y$
190 REM **INTRODUCIR EXTREMOS DE INTEGRACION**
200 PRINT AT 10,4;"EXTREMOS DE INTEGRACION ?"
210 PRINT : PRINT "EL VALOR DE X DEBE SER < QUE Y"
220 PRINT AT 15,0;"X=": INPUT XX
230 PRINT AT 15,3;XX
240 PRINT AT 15,16;"Y=": INPUT YY
250 PRINT AT 15,18;YY
260 IF XX>=YY THEN  GO TO 200
270 REM **INTRODUCIR NUMERO DE INTERVALOS **
```

```
280 INPUT "NUMERO DE INTERVALOS    ";N
290 REM ****************************************
300 REM ** COMIENZO DEL CALCULO DE LA INTEGRAL  **
310 REM ****************************************
320 REM ** D ES EL PASO O AMPLITUD DEL INTERVALO **
330 LET S=0: LET D=(YY-XX)/N
340 FOR I=1 TO N-1
350 LET T=XX+I*D
360 LET S=S+FN J(T)
370 NEXT I
380 LET S=D*(S+(FN J(XX)+FN J(YY))/2)
390 CLS
395 GO SUB 490
400 PRINT "LA INTEGRAL VALE    ";S
410 PRINT : PRINT : PRINT
420 PRINT "*** PULSE CUALQUIER TECLA PARA CONTINUAR  ***"
430 PAUSE 0
440 GO TO 10
450 REM
460 REM
470 REM
480 REM
490 REM ****************************************
500 REM ** REPRESENTACION GRAFICA DE LA FUNCION **
510 REM ****************************************
520 LET HD=(YY-XX)/255
530 LET MAX=-999999
540 FOR I=XX TO YY STEP HD
550 LET V=FN J(I)
560 IF ABS (V)>MAX THEN  LET MAX=ABS V
580 NEXT I
610 PLOT 0,87: DRAW 255,0
620 PLOT 0,87
630 FOR I=1 TO 255
640 LET A=FN J(XX+I*HD)
650 PLOT I,86+A*86/MAX
655 NEXT I
660 PRINT AT 12,0;xx: PRINT AT 12,28;yy
670 RETURN
```

Programa 2: Programa válido para Spectrum.

En el programa se utilizan tres variables para introducir los parámetros de la integración. Son:

- XX para el extremo inferior del intervalo
- YY para el extremo superior del intervalo
- N para el número de subintervalos de integración

Se utiliza una variable que se denomina T para calcular las áreas de los rectángulos formados por las sucesivas subdivisiones realizadas. En una variable que se denomina T se acumulan las áreas sucesivamente calculadas.

La representación gráfica de la función se realiza dentro de los límites impuestos por el tamaño de la pantalla. Por ello se ha utilizado un cambio de escala con el objeto de no sobrepasar estos límites

La última línea del programa principal supone una vuelta al principio. Si no se desea continuar con el programa habrá que provocar una interrupción de su ejecución.

PASATIEMPOS MATEMATICOS

Series de números

Se entiende que una sucesión es una serie de números que guardan una determinada relación entre ellos. A cada uno de los términos le sigue otro que se puede expresar en función del anterior gracias a la fórmula general de la sucesión.

Las series constituyen un buen entretenimiento. El determinar cuál es el número que sigue a una serie dada puede ser un serio quebradero de cabeza.

```
ELIGE GRADO DE DIFICULTAD (1 a 3)

    ********************************
    INTENTOS  1      ACIERTOS 1
    ********************************
¿QUIERES INTENTARLO CON OTRA SERIE? (S/N)

         CORRECTO

     25    36    49    64    81
¿CUAL SERIA EL SIGUIENTE NUMERO DE LA SERIE ? 100
```

Fig. 2.

```
10 REM **********************
20 REM * SERIES MATEMATICAS *
30 REM **********************
40 LET PUNTOS=0 : LET INTENTOS=0
50 CLS
60 REM * SERIES ALEATORIAS *
70 RANDOMIZE TIMER
80 LET INTENTOS=INTENTOS+1
90 LET C=0:LET A=0
100 LET I=INT (RND*3)+1:LET K=INT(RND*2)+2
110 LET R=INT(RND*5)+1
120 PRINT "ELIGE GRADO DE DIFICULTAD (1 a 3)"
130 LET J$=INKEY$:IF J$<"1" OR J$>"3" THEN GOTO 130
140 LET J=VAL(J$)
150 REM * GENERACION DE SERIES *
160 LET C=C+1
170 ON J GOSUB 500,600,700
180 LOCATE 10,C*5:PRINT A
190 LET I=I+K
200 IF C<5 THEN GOTO 160
210 REM * RESPUESTA *
220 LOCATE 12,4: PRINT "¿CUAL SERIA EL SIGUIENTE NUMERO DE LA SERIE"
230 LOCATE 12,48:INPUT S
240 ON J GOSUB 500,600,700
250 REM * RESULTADOS *
260 IF S=A THEN LOCATE 8,10:PRINT "CORRECTO" :PUNTOS=PUNTOS+1
270 IF S<>A THEN GOSUB 1000
280 LOCATE 3,5:PRINT"********************************"
290 LOCATE 4,5:PRINT "INTENTOS ";INTENTOS;"    ACIERTOS";PUNTOS
300 LOCATE 5,5:PRINT"********************************"
310 REM * OTRO JUEGO *
320 PRINT "¿QUIERES INTENTARLO CON OTRA SERIE? (S/N)"
330 V$=INKEY$: IF V$="" THEN GOTO 330
340 IF V$="S" OR V$="s" THEN GOTO 50
350 END
500 REM * SERIE DIFICULTAD 1 *
510 LET A=I*R
520 RETURN
600 REM * SERIE DIFICULTAD 2 *
610 LET A=A+I
620 RETURN
700 REM * SERIE DIFICULTAD 3 *
710 LET A=R^K
720 LET R=R+1
730 RETURN
1000 REM * RESPUESTA INCORRECTA *
1010 LOCATE 8,10:PRINT "INCORRECTO"
1020 LOCATE 9,10:PRINT "EL NUMERO ERA:";A
1030 RETURN
```

*Programa 3: Programa válido para IBM-PC. En el caso de Amstrad y MSX elimine linea 70 y sustituya las lineas en las que aparezca LOCATE F,C por LOCATE C,F. Para Spectrum sustituirlo por PRINT AT F,C y sustituir linea 170 por GOSUB Z añadiendo linea 165 LET Z=J*100+400.*

```
ELIGE GRADO DE DIFICULTAD (1 a 3)

    ********************************
    INTENTOS  2     ACIERTOS 2
    ********************************
¿QUIERES INTENTARLO CON OTRA SERIE? (S/N)

         CORRECTO

    1    5    12   22   35

 ¿CUAL SERIA EL SIGUIENTE NUMERO DE LA SERIE ? 51
```

Fig. 3.

```
ELIGE GRADO DE DIFICULTAD (1 a 3)

    ********************************
    INTENTOS  3     ACIERTOS 3
    ********************************
¿QUIERES INTENTARLO CON OTRA SERIE? (S/N)

         CORRECTO

    3    12   21   30   39

 ¿CUAL SERIA EL SIGUIENTE NUMERO DE LA SERIE ? 48
```

Fig. 4.

Simulación de trayectorias parabólicas

Cuando se tira una piedra, o cuando se da una patada a un balón sucede el mismo fenómeno que el del lanzamiento de un proyectil. El camino que recorren todos estos objetos se denomina TRAYECTORIA PARABOLICA.

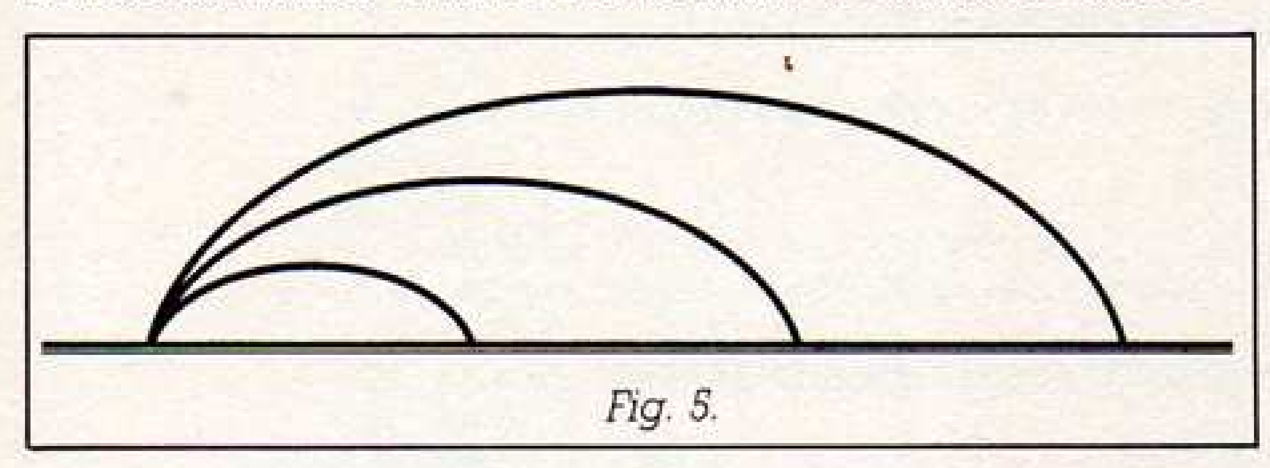


Fig. 5.

En Matemáticas se denomina **parábola** a la curva que describe este tipo de movimiento. Su estudio físico correspondiente se aplica a los fenómenos de tiro parabólico.

Si analizamos cualquiera de los dos experimentos anteriores observaremos que la piedra y el balón llegarán a una mayor altura en función de la inclinación que demos al tiro, es decir, dependen del **ángulo de lanzamiento.** Asimismo, si queremos alcanzar un objetivo concreto, como podría ser una portería de fútbol, deberíamos dar una mayor o menor velocidad al balón. Cuanto mayor sea la velocidad, mayor será la distancia alcanzada.

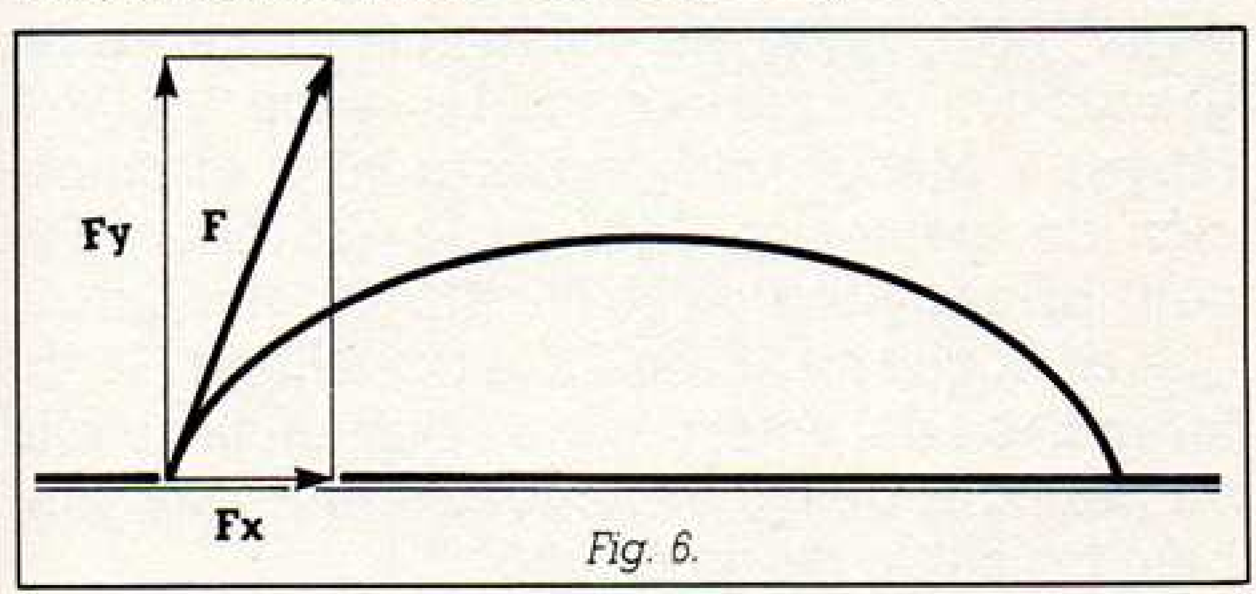


Fig. 6.

En definitiva, jugamos con dos variables:

— Angulo de inclinación.
— Velocidad inicial del objeto.

Dependiendo de estas dos variables variarán:

— Distancia alcanzada.
— Altura máxima.

Por supuesto, en todo este tipo de fenómenos existe una variable adicional que interviene en todo momento. Esta variable hace que la piedra caiga al suelo y no permite que siga con su movimiento ascendente. Se denomina **gravedad** y su valor es constante e igual a 9.8 metros/segundo.

Conclusiones

En fenómenos tales como el lanzamiento de una piedra, el disparo de un proyectil o el movimiento de un balón al ser golpeado por un pie intervienen varias fuerzas:

— Una fuerza favorable que es la ejercida por dispositivo de lanzamiento (mano, pié, cañón).

— Una fuerza desfavorable que es la de la gravedad.

Dado que la fuerza de lanzamiento es un vector, la magnitud de ésta dependerá del ángulo correspondiente.

La distancia y la altura máxima alcanzadas son función del balance final de estas fuerzas contrarias. Este balance determina una trayectoria de movimiento de tipo parabólico.

```
10 REM ** TIRO PARABOLICO **
20 INPUT "VELOCIDAD INICIAL (EN M/SEG):";V
30 IF V<=0 THEN GOTO 20
40 INPUT "ANGULO LANZAMIENTO (EN GRADOS):";A
50 IF A>=90 THEN PRINT AT 21,0;"ANGULO INCORRECTO": PAUSE 50:CLS:GOTO 40
60 LET G=9.8
70 LET A=(A*2*PI)/360
80 LET T1=(V*SIN (A))/G
90 LET Y1=((V^2)*(SIN (A))/(2*G)
100 PRINT "ALTURA MAXIMA DEL PROYECTIL:"
110 PRINT TAB 9;Y1:PRINT "EN T=";T1;" SEGUNDOS "
120 LET T2=2*T1
130 LET X1=(V^2)*SIN (2*A)/G
140 PRINT: PRINT: PRINT
150 PRINT "DISTANCIA MAXIMA ALCANZADA:"
160 PRINT TAB (9);X1: PRINT "EN T=";T2;" SEGUNDOS"
170 PRINT: PRINT: PRINT
180 PRINT TAB (3);"PARA VISUALIZAR TRAYECTORIA": PRINT TAB 8;"DEL PROYECTIL"
190 PRINT: PRINT
```

```
200 PRINT TAB (5); FLASH,1;"PULSE CUALQUIER TECLA"
210 PAUSE 0
220 CLS
230 LET N=1
240 IF Y1>X1 THEN GOTO 270
250 LET N=X1/255
260 GOTO 280
270 LET N=Y1/175
280 PLOT 0,175: DRAW 0,-175: DRAW 255,0
290 LET M2=0
300 FOR X=0 TO X1 STEP X1/40
310 LET Y=(X*TAN (A))-.5*(G*(X^2))/((V*COS (A))^2)
320 IF (Y/N)>M2 THEN LET M2=Y/N
330 PLOT X/N,Y/N
340 NEXT X
350 LET M1=X1/(2*N)
360 IF X1>=1000 THEN GOTO 430
370 PLOT (X1/N),0: DRAW 0,8: PRINT AT 20,((31*X1/N)/255)-8;X1;"m."
380 IF Y1>=1000 THEN GOTO 450
390 PLOT 0,M2: DRAW 8,0: PRINT AT ((-21*M2/175)+21),1;Y1;"m."
400 PLOT 0,0: DRAW INK 2;0,M2
410 PLOT 0,0: DRAW INK 2;2*M1,0
420 GOTO 9999
430 PLOT (X1/N),0: DRAW 0,8: PRINT AT 20,((31*X1/N)/255)-8;X1/1000;"Km."
440 GOTO 380
450 PLOT 0,M2: DRAW 8,0: PRINT AT ((-21*M2/175)+21),1;Y1/1000;"Km."
460 GOTO 400
```

Programa 4: Programa válido para Spectrum.

El programa se divide en dos partes. En la primera se calculan la altura máxima y la distancia alcanzadas en función de la velocidad inicial y del ángulo de lanzamiento, que son dos datos que se introducen al principio del programa. El ángulo no puede ser mayor de 90 grados. En cualquier caso el programa no lo permite.

En la segunda parte del programa se dibuja la trayectoria descrita por el objeto lanzado. Dado que se manejan comandos gráficos el programa variará sustancialmente en función del tipo de ordenador que se emplee. Por ello recomendamos que vea el cuadro adjunto antes de introducir el programa.

PRUEBA TUS CONOCIMIENTOS

Tabla periódica de los elementos químicos

Las propiedades químicas de un elemento dependen de su configuración electrónica. Se suelen agrupar todos los elementos con estructuras electrónicas semejantes. Al conjunto de todos los elementos químicos agrupados por sus configuraciones electrónicas se le denomina **Tabla periódica.**

Mucho antes de que se estableciese una clasificación perfecta de los elementos, los químicos se habían dado cuenta de la periodicidad de algunas propiedades. La primera experiencia se debe a Döbereiner, que en 1817 separó en grupos de tres alguno de los elementos, con la peculiaridad de que el elemento central tenía un peso atómico que era igual a la media aritmética de los tres. Más adelante, en 1862, Chancourtois estableció una clasificación de los elementos en forma de hélice. En 1868, Newlans enunció su "Ley de octavas", ordenando los elementos químicos en grupos con una periodicidad de ocho. En 1869, Meyer y Mendelejeff publicaron su clasificación de elementos. Más tarde Werner y Paneth presentaron su sistema periódico largo, que es el que se utiliza en la actualidad. Se diferenciaba del anterior en la existencia no sólo de períodos de ocho elementos, sino que también los había de dieciocho.

Los elementos que forman parte de un mismo grupo tienen una serie de propiedades semejantes, como son:

- Configuración electrónica.
- Potencial de ionización.
- Afinidad electrónica.
- Electronegatividad.
- Volumen atómico.

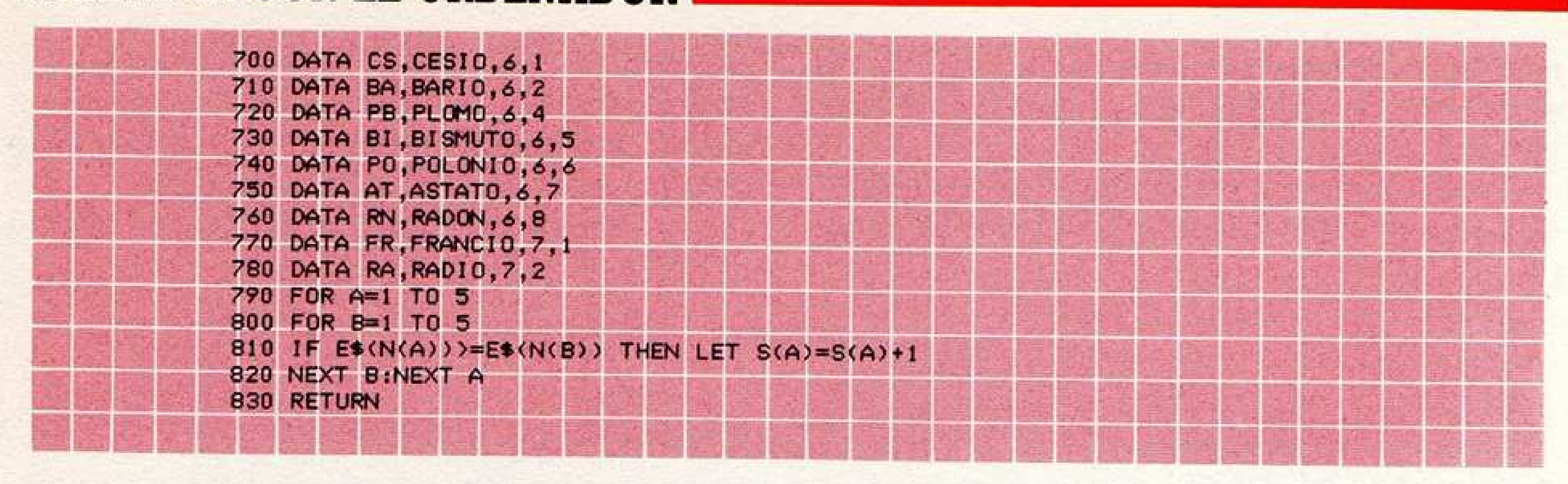
En general las reacciones de elementos de un mismo grupo de la tabla periódica con otros compuestos se realizan de modo semejante. Esta característica es importante, puesto que dada una reacción química de un compuesto con un elemento podemos en cierto

modo prever las reacciones que el resto de los elementos del grupo van a tener con el mismo compuesto.

El objeto del programa es el de realizar un pequeño test sobre una tabla periódica simplificada por problemas de espacio en la pantalla de algunos ordenadores. En sucesivos tomos iremos ampliando las posibilidades de programación que la química permite.

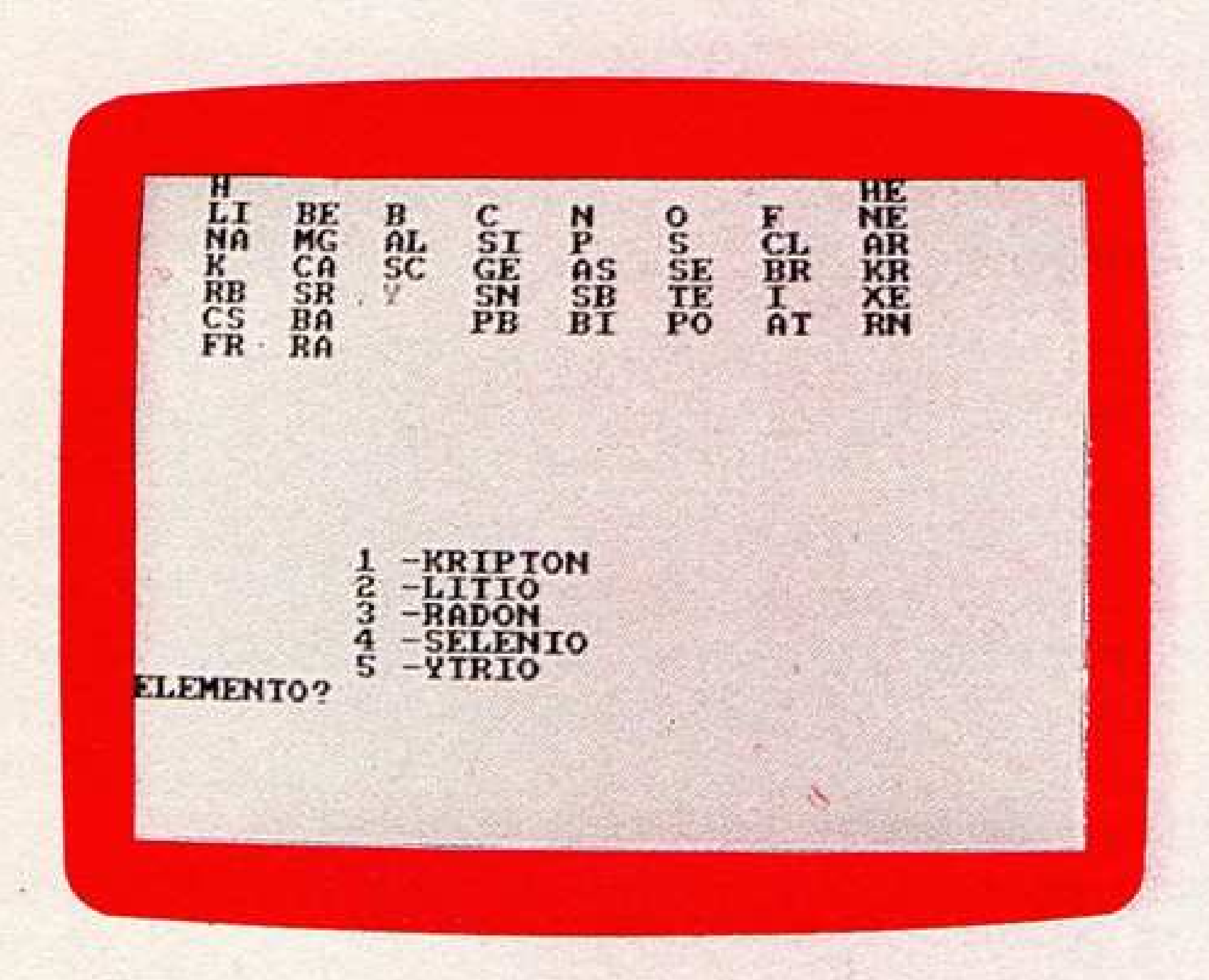
Este es un programa del tipo de test de

```
10 MODE 1
20 PAPER 2:INK 0,0
30 CLS
40 DIM S$(43):DIM E$(43):DIM F(43):DIM C(43)
50 FOR I=1 TO 43
60 READ S$(I),E$(I),F(I),C(I)
70 NEXT I
80 FOR I=1 TO 43
90 LOCATE C(I)*4,F(I):PRINT S$(I)
100 NEXT I
110 FOR I=1 TO 5
120 LET N(I)=INT(RND*43)+1
130 NEXT I
140 FOR H=1 TO 5
150 FOR I=1 TO 5
160 IF H=I THEN GOTO 180
170 IF N(H)=N(I) THEN N(I)=INT(RND*43)+1:GOTO 170
180 NEXT I
190 NEXT H
200 INK 3,2,18
210 LOCATE C(N(1))*4,F(N(1)):PEN 3:PRINT S$(N(1))
220 PEN 0
230 GOSUB 790
240 LOCATE 1,15
250 FOR I=1 TO 5
260 FOR J=1 TO 5
270 IF S(J)=I THEN LOCATE 10,I+14:PRINT I;"-";E$(N(J))
280 NEXT J
290 NEXT I
300 LOCATE 1,20:PRINT "ELEMENTO?"
310 LET E$=INKEY$:IF E$="" THEN GOTO 310
320 LET E=VAL(E$)
330 IF E=S(1) THEN PEN 3:PRINT "ACERTASTE":PEN 0
340 IF E<>S(1) THEN PEN 3:PRINT "FALLASTE INTENTALO DE NUEVO":PEN 0:GOTO 310
350 END
360 DATA H,HIDROGENO,1,1
370 DATA HE,HELIO,1,8
380 DATA LI,LITIO,2,1
390 DATA BE,BERILIO,2,2
400 DATA B,BORO,2,3
410 DATA C,CARBONO,2,4
420 DATA N,NITROGENO,2,5
430 DATA O,OXIGENO,2,6
440 DATA F,FLUOR,2,7
450 DATA NE,NEON,2,8
460 DATA NA,SODIO,3,1
470 DATA MG,MAGNESIO,3,2
480 DATA AL,ALUMINIO,3,3
490 DATA SI,SICILIO,3,4
500 DATA P,FOSFORO,3,5
510 DATA S,AZUFRE,3,6
520 DATA CL,CLORO,3,7
530 DATA AR,ARGON,3,8
540 DATA K,POTASIO,4,1
550 DATA CA,CALCIO,4,2
560 DATA SC,ESCANDIO,4,3
570 DATA GE,GERMANIO,4,4
580 DATA AS,ARSENICO,4,5
590 DATA SE,SELENIO,4,6
600 DATA BR,BROMO,4,7
610 DATA KR,KRIPTON,4,8
620 DATA RB,RUBIDIO,5,1
630 DATA SR,ESTRONCIO,5,2
640 DATA Y,YTRIO,5,3
650 DATA SN,ESTANO,5,4
660 DATA SB,ANTIMONIO,5,5
670 DATA TE,TELURO,5,6
680 DATA I,IODO,5,7
690 DATA XE,XENON,5,8
```



```
700 DATA CS,CESIO,6,1
710 DATA BA,BARIO,6,2
720 DATA PB,PLOMO,6,4
730 DATA BI,BISMUTO,6,5
740 DATA PO,POLONIO,6,6
750 DATA AT,ASTATO,6,7
760 DATA RN,RADON,6,8
770 DATA FR,FRANCIO,7,1
780 DATA RA,RADIO,7,2
790 FOR A=1 TO 5
800 FOR B=1 TO 5
810 IF E$(N(A))>=E$(N(B)) THEN LET S(A)=S(A)+1
820 NEXT B:NEXT A
830 RETURN
```

Programa 5: Programa válido para Amstrad y MSX. Para IBM-PC sustituir LOCATE C,F por LOCATE F,C.



varias posibilidades entre las cuales debemos elegir una. Cuando se ejecute, el ordenador presentará en pantalla la tabla periódica y uno de los elementos especialmente iluminado. Además, aparecen cinco nombres de elementos químicos de los cuales sólo uno es el correcto.

El programa ha sido estructurado en las siguientes partes:

- Lectura de datos.
- Visualización de la tabla.
- Visualización de un elemento de la tabla diferenciado.
- Visualización de las alternativas del test.

Para la lectura de los datos se han empleado cuatro conjuntos de variables con un subíndice. El primero de ellas sirve para almacenar el símbolo del elemento químico. El segundo guarda el nombre del elemento. Los otros dos sirven para las coordenadas de posición de la tabla en la pantalla, es decir, son los números de fila y de columna que ocupa cada elemento.

La elección del elemento en cada pregunta se produce de forma aleatoria. Los elementos que sean las alternativas no correctas en el test se eligen también de esta manera. Para ello se utiliza otro conjunto de variables con subíndice denominado N(5), en el que el primer elemento del conjunto coincide con la respuesta verdadera. Para evitar la posibilidad de que ocurran repeticiones de los elementos en alguna de las alternativas del test, se comprueba que los cinco números extraídos al azar no sufran repeticiones. En el caso de que así sucediese, se volvería a extraer al azar aquel número que coincidiese.

Dado que se ha seguido como criterio el asignar como verdadero el primer elemento de los cinco que se extraen al azar podría parecer que la respuesta al test sería tan fácil como contestar que el primer elemento es el cierto siempre. Para evitarlo se ordenan alfabéticamente los cinco posibles elementos. Se ha empleado el método de ordenación por indexación. Consiste en poner un número de orden delante de cada elemento en función del puesto alfabético que ocupen. De esta forma no se ordena el conjunto, sino que se le asignan unos números que luego serán claves a la hora de visualizarlos.

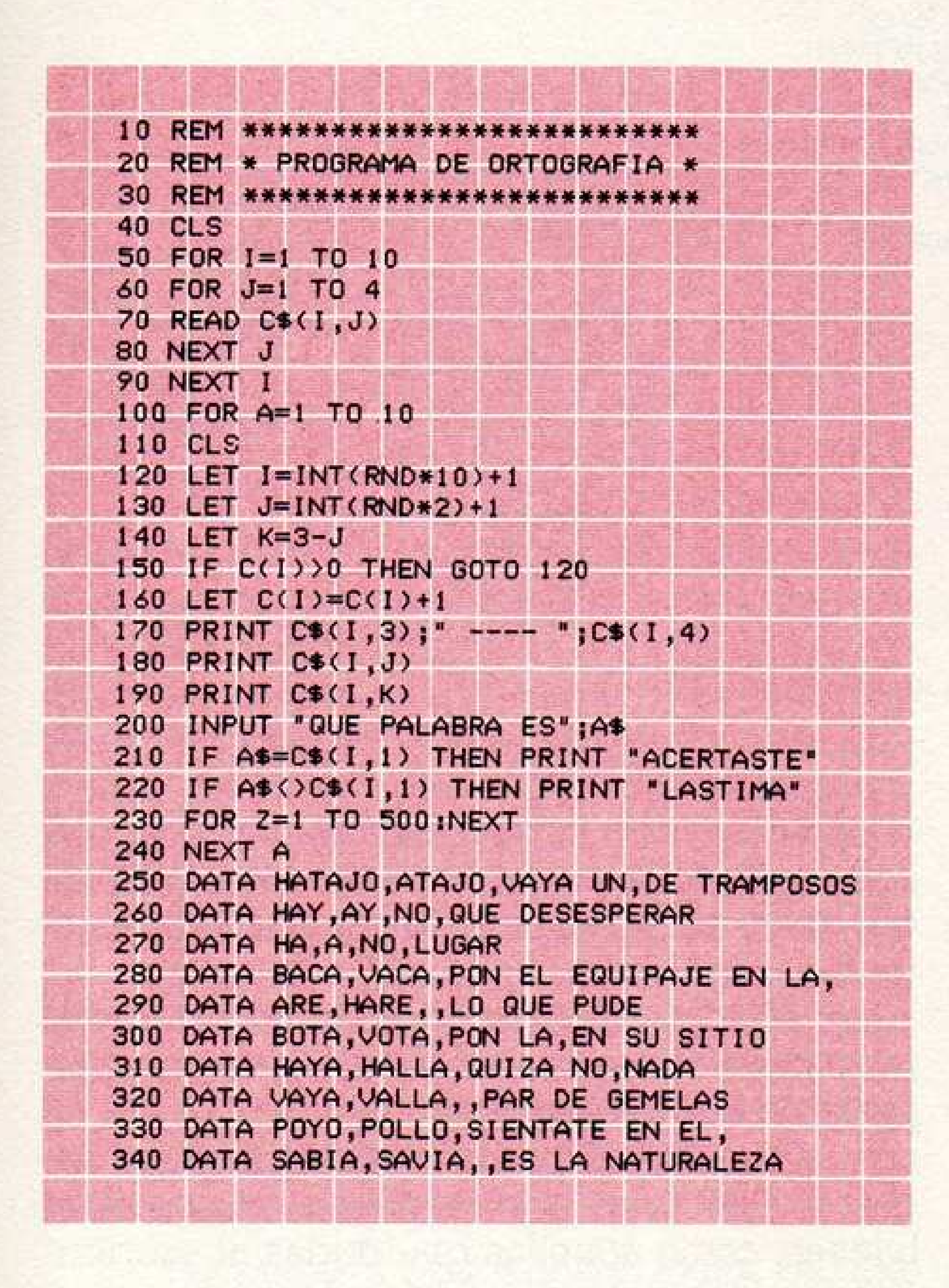
Sugerencias

Además de un programa del tipo test puede construirse una segunda alternativa en la que el ordenador pregunte el símbolo de un elemento. El programa podría ampliarse tam-

bién a una serie de propiedades de los elementos como son las valencias, el peso atómico, etc.

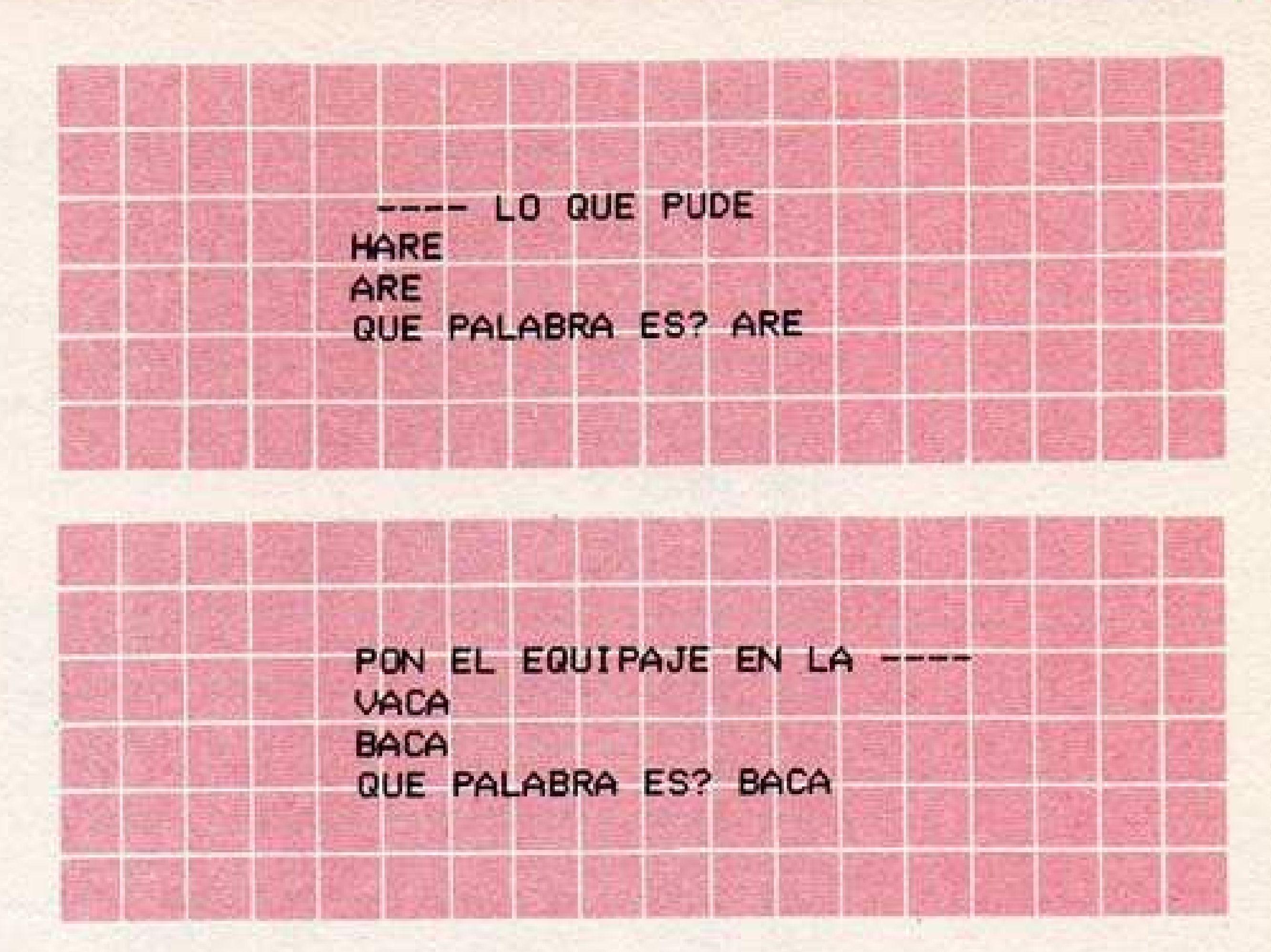
Test de Semántica

El siguiente programa trata de probar la capacidad de dos palabras de igual sonido pero distinta ortografía. Deberá elegirse entre dos alternativas de las que sólo una será correcta.



```
10 REM **************************
20 REM * PROGRAMA DE ORTOGRAFIA *
30 REM **************************
40 CLS
50 FOR I=1 TO 10
60 FOR J=1 TO 4
70 READ C$(I,J)
80 NEXT J
90 NEXT I
100 FOR A=1 TO 10
110 CLS
120 LET I=INT(RND*10)+1
130 LET J=INT(RND*2)+1
140 LET K=3-J
150 IF C(I)>0 THEN GOTO 120
160 LET C(I)=C(I)+1
170 PRINT C$(I,3);" ---- ";C$(I,4)
180 PRINT C$(I,J)
190 PRINT C$(I,K)
200 INPUT "QUE PALABRA ES";A$
210 IF A$=C$(I,1) THEN PRINT "ACERTASTE"
220 IF A$<>C$(I,1) THEN PRINT "LASTIMA"
230 FOR Z=1 TO 500:NEXT
240 NEXT A
250 DATA HATAJO,ATAJO,VAYA UN,DE TRAMPOSOS
260 DATA HAY,AY,NO,QUE DESESPERAR
270 DATA HA,A,NO,LUGAR
280 DATA BACA,VACA,PON EL EQUIPAJE EN LA,
290 DATA ARE,HARE,,LO QUE PUDE
300 DATA BOTA,VOTA,PON LA,EN SU SITIO
310 DATA HAYA,HALLA,QUIZA NO,NADA
320 DATA VAYA,VALLA,,PAR DE GEMELAS
330 DATA POYO,POLLO,SIENTATE EN EL,
340 DATA SABIA,SAVIA,,ES LA NATURALEZA
```

Programa 6: Programa válido para Amstrad e IBM-PC. Para MSX y Commodore ver tabla de equivalencia y sustituir líneas 120 y 130. Para Spectrum incluir línea 45 DIM C$(10,4,15) y poner todos los datos entre comillas.



```
---- LO QUE PUDE
HARE
ARE
QUE PALABRA ES? ARE
```

```
PON EL EQUIPAJE EN LA ----
VACA
BACA
QUE PALABRA ES? BACA
```

PARA LOS MAS PEQUEÑOS

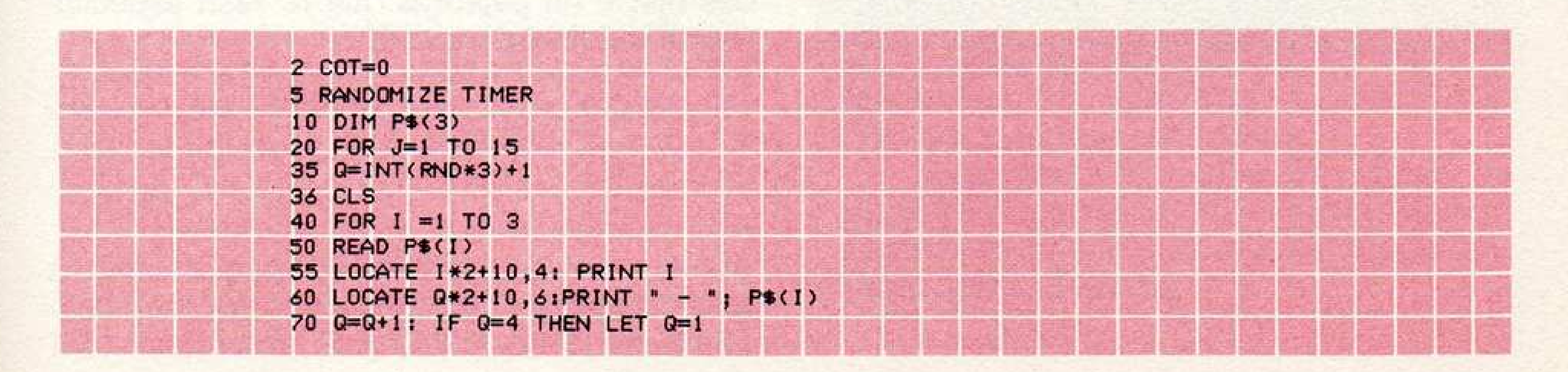
Test de ortografía sencilla

La ortografía es quizá uno de los campos donde mayores problemas pueden encontrarse. ¿Quién no ha confundido una "b" con una "v", una "g" con una "j" u omitido una "h"?

Los métodos tradicionales de aprendizaje de ortografía consistían en memorizar una serie de reglas que no siempre respondían a un patrón lógico. Consideramos que esos métodos no son los más eficaces. Mediante el programa de ortografía que viene a continuación, pretendemos que el aprendizaje de esta materia se lleve a cabo de una forma divertida. En cualquier caso te proponemos un método de aprender ortografía, que es el siguiente:

— Lee todo lo que puedas.
— Fíjate bien en las palabras.
— Busca en un diccionario aquéllas cuyo significado no conozcas.

El siguiente programa servirá para conocer la ortografía de una serie de palabras de las cuales puede que desconozcas la forma correcta de escritura.

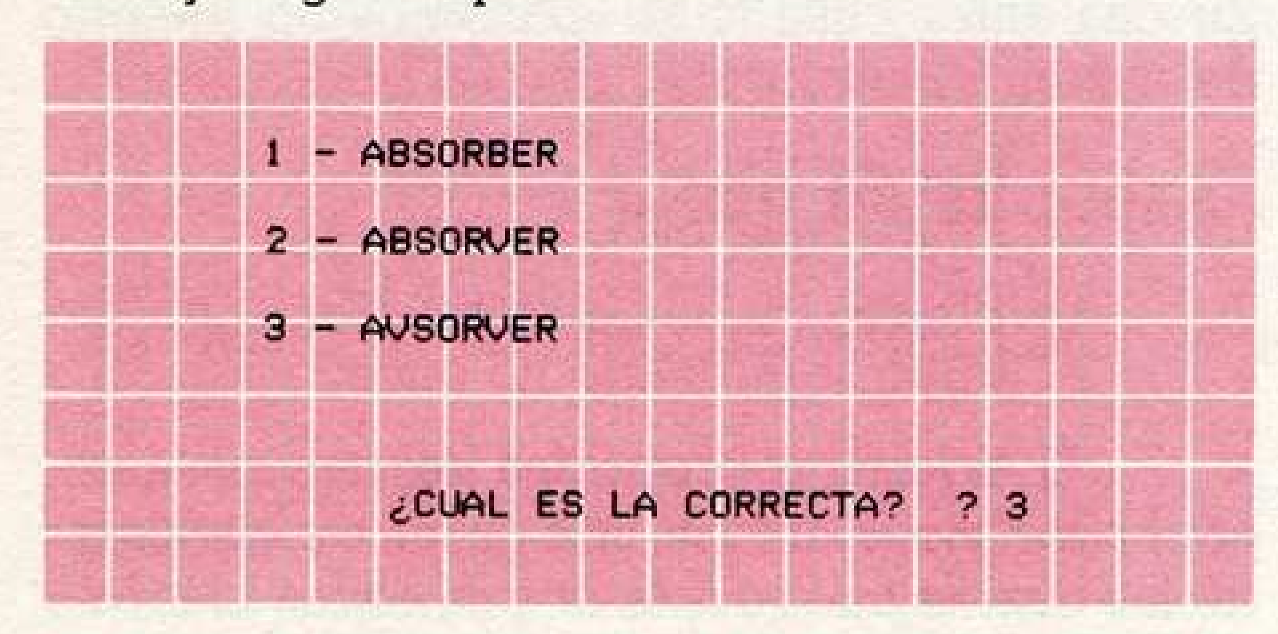


```
2 COT=0
5 RANDOMIZE TIMER
10 DIM P$(3)
20 FOR J=1 TO 15
35 Q=INT(RND*3)+1
36 CLS
40 FOR I =1 TO 3
50 READ P$(I)
55 LOCATE I*2+10,4: PRINT I
60 LOCATE Q*2+10,6:PRINT " - "; P$(I)
70 Q=Q+1: IF Q=4 THEN LET Q=1
```

```
90 NEXT I
100 LOCATE 20,10: PRINT "¿CUAL ES LA CORRECTA?"
110 LOCATE 20,33:INPUT R
120 IF R>3 OR R<1 THEN GOTO 110
130 IF R=Q THEN LET COT=COT+1
135 LOCATE 1,2:PRINT"**********************************************"
140 LOCATE 2,4:PRINT"TUS ACIERTOS SON ";COT;"   TUS INTENTOS SON ";J
145 LOCATE 3,2:PRINT"**********************************************"
146 FOR K=1 TO 1000 :NEXT K
150 NEXT J
160 IF COT=15 THEN BEEP
170 IF COT<5 THEN GOSUB 1000
500 DATA "ABSORBER","ABSORVER","AVSORVER"
510 DATA "BOVEDA","VOBEDA","BOBEDA"
520 DATA "VIBORA","VIVORA","BIVORA"
530 DATA "HERVIR","ERVIR","HERBIR"
550 DATA "RECIBIR","RECIVIR","REZIVIR"
560 DATA "SORBER","SORVER","SORBEL"
570 DATA "MUGIR","MUJIR","MUGUIR"
580 DATA "HEXAGONO","EXAGONO","HESAGONO"
590 DATA "HERMETICO","ERMETICO","HELMETICO"
600 DATA "ALMOHADA","ALMOADA","HALMOADA".
610 DATA "GARAJE","GARAGE","JAREJE"
620 DATA "ORTOPEDIA","HORTOPEDIA","ORTHOPEDIA"
630 DATA "ERMITA","HERMITA","EHRMITA"
640 DATA"POLVO","PLOVO","POLBO"
650 DATA "OBSERVAR","OVSERVAR","EVSERBAR"
1000 PRINT"                                   *        *"
1010 PRINT"                                   **      **"
1020 PRINT"                                  ***     ***"
1030 PRINT"                                   ** *** **"
1040 PRINT"                                    *      *"
1050 PRINT"                                   *  O  O  *"
1060 PRINT"                                   *        *"
1070 PRINT"                                   *  /oo\  *"
1080 PRINT"                                    * \__/ *"
1090 PRINT"                                      ****"
1105 RETURN
```

Programa 7: Programa válido para IBM-PC.

El ordenador muestra en pantalla tres formas de escribir una palabra. Fíjate bien en ellas y elige la que creas correcta.



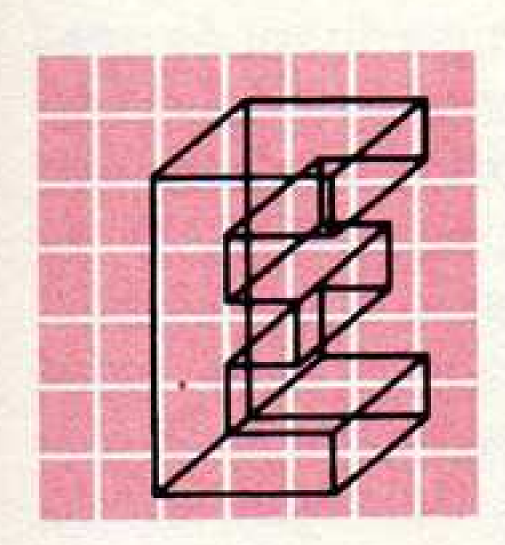
Si no estás seguro, intenta recordar una frase en la que hayas visto escrita la palabra.

Si respondes con la opción correcta, el contador de aciertos ganará un punto. Contesta con cuidado, porque si fallas muchas palabras no le va a gustar al ordenador. Si lo haces bien el ordenador se pondrá muy contento.

Comprueba cómo andan en ortografía tus amigos, jugando con el programa y comparando los marcadores.

Si estas palabras te resultan fáciles o llegas a conocerlas perfectamente, busca otras nuevas, como aquellas que dudas al escribir. Puedes introducirlas en el programa sustituyendo las sentencias DATA, recordando que la palabra correcta es la que debes poner la primera y que las otras dos serán palabras que se parezcan pero que sean incorrectas. Puedes fijarte en palabras que te sean especialmente difíciles de escribir.

PEQUEÑA HISTORIA DE LA INFORMATICA



L siguiente avance en nuestra materia se realiza ya fuera de la vieja Europa. Herman Hollerith se presenta a un concurso para mejorar la elaboración de censos en la Oficina Censal de los Estados Unidos. El problema de la elaboración de los censos era arduo, y no llegaba a completarse cuando ya era necesario volver a comenzar de nuevo. Hollerith tomó como modelo las tarjetas perforadas de los telares de Jacquard y observó que muchas preguntas del censo sólo tenían dos respuestas. Incluso aquellos casos en los que las respuestas podían ser algo más numerosas, el problema se podría resolver codificándolas, es decir, realizando orificios en determinadas posiciones. Además, estos orificios, (nuevo avance) podrían detectarse por medios eléctricos, no mecánicos. El avance que supuso la tarjeta de Hollerith fue enorme, y lo trataremos con detenimiento más adelante.

Sin embargo, para que el lector recapacite sobre su utilidad, diremos que el censo de los EE.UU. de 1890 contenía los datos de 62 millones de personas, y tardó en realizarse dos años. El censo anterior, el de 1880, tardó siete años y medio en llevarse a cabo.

La revista "Scientific Américan" se interesó en el nuevo proceso de la información y dedicó gran parte de su número de agosto de 1890 (incluyendo la portada) a la obra de Hollerith.

Nuestro inventor, paralelamente, fundó su fábrica para dedicarse a la comercialización de las máquinas tabuladoras (además de listo, era, desde luego, muy práctico). Se llamó Tabulating Machine Company, TMC, y tras varias fusiones con otras compañías, llegó a formar parte, en 1924, de la International Business Machines. Pero la TMC no fue la única empresa dedicada a comercializar este tipo de máquinas. Hollerith además de "inventar" la ficha perforada fue uno de los pioneros del "sistema de alquiler", que tan buenos resultados ha dado hasta nuestros días (Compañía Telefónica, máquina tragaperras, etc.); y en 1905 la Oficina del Censo decidió convocar un concurso para el desarrollo de una nueva máquina y liberarse de los alquileres de la TMC. El concurso lo ganó James Powers, que más tarde, en 1911, fundó su propia empresa, Powers Accounting Machine Company. Esta úl-

Sabía usted que...

¿Sabe usted quién «inventó» el microprocesador?

El "inventor del chip", Ted Hoff, tenía cuarenta y cinco años cuando decidió retirarse a investigar sobre algunas ideas que tenía, y de cuyo interés comercial no estaba seguro.

El resultado de sus trabajos es el microprocesador. En 1969 los microprocesadores no existían; en 1975 había un censo de unos 750.000; en la actualidad se calcula que funcionan unos 100 millones, y para fin de siglo los oráculos más conservadores indican que habrá unos 1.000 millones.

Evidentemente, ni Ted Hoff ni la compañía Intel que los comercializó, pudo ni siquiera soñar la repercusión de su invento (aunque sólo considerásemos el aspecto comercial).

tima (como Tabulating Machines Division), pasó a formar parte de la Remington Rand Corp., fabricante del primer ordenador comercial, el UNIVAC I. (Debemos resaltar la diferencia entre crear un ordenador (prototipo) para una Universidad u organismo público, y fabricar un determinado ordenador para comercializarlo).

Una de las ciencias que se desarrolló enormemente gracias a este tipo de cálculos que utilizaban tarjetas de Hollerith fue la Astronomía clásica. Los resultados (se manejaba un número enorme de datos) eran espectaculares y este tipo de cálculos astronómicos se han venido realizando casi sin cambios hasta hace muy pocos años.

Sabía usted que...

¿Sabía usted que la velocidad de cómputo de los ordenadores actuales es un millón de veces superior a la de los primeros ordenadores de los años 50?

Los primeros ordenadores tenían un enorme volumen. Veamos algunos ejemplos: un ordenador de nuestros días, que ocupa el volumen de una pequeña máquina de escribir tendría la misma capacidad de cómputo que otro de los años 50 que ocupara totalmente un apartamento de 50 metros cuadrados.

En el año 1965 nuestro ordenador tendría el tamaño de una caravana no muy pequeña; en 1975, el tamaño de un televisor doméstico, y ¿dónde llegaremos impulsados por este espíritu de jibarización que nos anima?

Pero fue en Cambridge (Massachusetts) donde se dio el siguiente avance importante en las máquinas de calcular automáticas, como evolución de las ideas concebidas por Babbage más de cien años atrás, en el otro Cambridge (Inglaterra).

En esta población americana vio la luz la primera máquina realmente automática, ciento doce años más tarde que su predecesora de Cambridge (Inglaterra).

En 1937, en la Universidad de Harvard, Howard Aiken aprovechó las técnicas de las máquinas anteriores para crear una máquina totalmente automática. Sin embargo, para ello tuvo que recabar la ayuda de la mayor fábrica de máquinas que utilizaban las tarjetas perforadas: la International Business Machines Corporation (IBM). La máquina fue ofrecida a la Universidad. Su nombre era ASCC (Automatic Sequence Controlled Calculator), pero fue conocida como Harvard Mk1. Esta máquina constituye un prototipo antecesor de los ordenadores actuales. Sus unidades de entrada y salida de datos eran bastante buenas: utilizaban las tarjetas perforadas para la introducción de los datos, y los resultados podían obtenerse también en tarjetas o bien eran mecanografiados. Era una máquina muy completa, que había sido fabricada sin economías de ningún tipo en tiempo, y dinero, e incluso volumen. Estuvo funcionando muchos años en Harvard ininterrumpidamente.

Sin embargo, poco tiempo después apareció otra máquina cuyos avances respecto de la anterior eran considerables; se trataba de la primera máquina electrónica: el ENIAC. El equipo fue ultimado en 1946, sólo dos años más tarde que el Harvard Mk1. Este primer ordenador o computador fue creado por J. P. Eckert y J. P. Mauchly de la Universidad de Pennsylvania. Desgraciadamente, es frecuente que las guerras y los objetivos militares supongan enormes avances en la ciencia. El ENIAC, por ejemplo, fue desarrollado y subvencionado por el Ejército Americano, y se diseñó para que resolviera fundamentalmente problemas militares (trayectorias, etc.). Era una máquina electrónica, es decir, las secuencias de operaciones se llevaban a cabo utilizando circuitos electrónicos. Evidentemente, la velocidad en la realización de los cálculos aumentó muchísimo, pero las válvulas que se usaban en los circuitos se estropeaban con frecuencia (eran muchísimas) y resultaba muy complicado sustituirlas. Hoy en día las válvulas han sido remplazadas por transistores, que son mucho más fiables, y de un volumen mucho menor.

Pero ¿no nos hemos olvidado del sistema utilizado para el cómputo y almacenamiento de los datos? He aquí una cuestión importante. Hasta el momento, todas las máquinas de las que hemos hablado utilizaban un sistema decimal para representar y almacenar los datos. El sistema binario, sin embargo, comenzó a usarse, o al menos a considerarse, a partir del momento en el que se empezaron a utilizar válvulas, es decir, hacia 1919. A partir de 1929, la American General Election se había decidido a utilizar una válvula thyratron como relé, haciendo posible la acumulación de los datos. La idea no es fruto de un único inven-

tor, ya que normalmente cada trabajo de investigación suele apoyarse o partir de principios que han sido desarrollados por otro investigador anterior. En 1931, en Inglaterra, William Philips publicó un trabajo en el que propugnaba la utilización de esta tecnología que utilizaba válvulas y un sistema binario.

Sin embargo, la persona que implantó realmente el uso del sistema binario fue John von Neumann. El fue el pionero en el diseño de una arquitectura que sería la utilizada por la industria durante muchísimo tiempo. Su aportación fue muy importante, ya que al resolver el problema de poder mantener en el ordenador instrucciones y datos a un mismo tiempo, ya no sería necesario en lo sucesivo establecer lenta y laboriosamente las tareas desde el exterior y además, el ordenador podía tomar decisiones lógicas internamente.

En agosto de 1954, el equipo ENIAC convocó una conferencia en la Moore School. Esta conferencia tuvo extraordinaria importancia, ya que se reunían grandes especialistas en cálculo y computación para discutir un nuevo diseño, el EDVAC. (De hecho, la mayoría de los participantes en esa conferencia acabaron siendo contratados como consultores en las principales empresas de ordenadores.) Pensemos, por ejemplo, en la Remington Rand (luego Sperry Rand, posteriormente UNIVAC-Sperry, y hoy UNISYS), que contrató a Eckert, y Mauchly; o en la IBM, que contrató a von Neumann y Goldstine).

La IBM, consorcio de varias empresas, había sido creada en 1911 bajo el nombre de CTRC (Computing, Tabulating and Recording Company-Compañía de Computación, Tabulación y Registro), y más tarde, en 1924 cambió su nombre al que ahora tiene. De la IBM, el coloso multinacional, hablaremos más adelante, pero digamos aquí que la enorme empresa había tratado por todos los medios de participar en la Conferencia. No le fue posible, aunque estaba muy relacionada en proyectos militares. Y aquí aparece otro personaje controvertido: Alan Turing. Este investigador tenía gran confianza en sí mismo, hasta el punto de que después de rechazar el trabajo como ayudante de von Newman, llegó incluso a decir ni más ni menos que él era siempre capaz de mejorar cualquier proyecto realizado por otro investigador cualquiera.

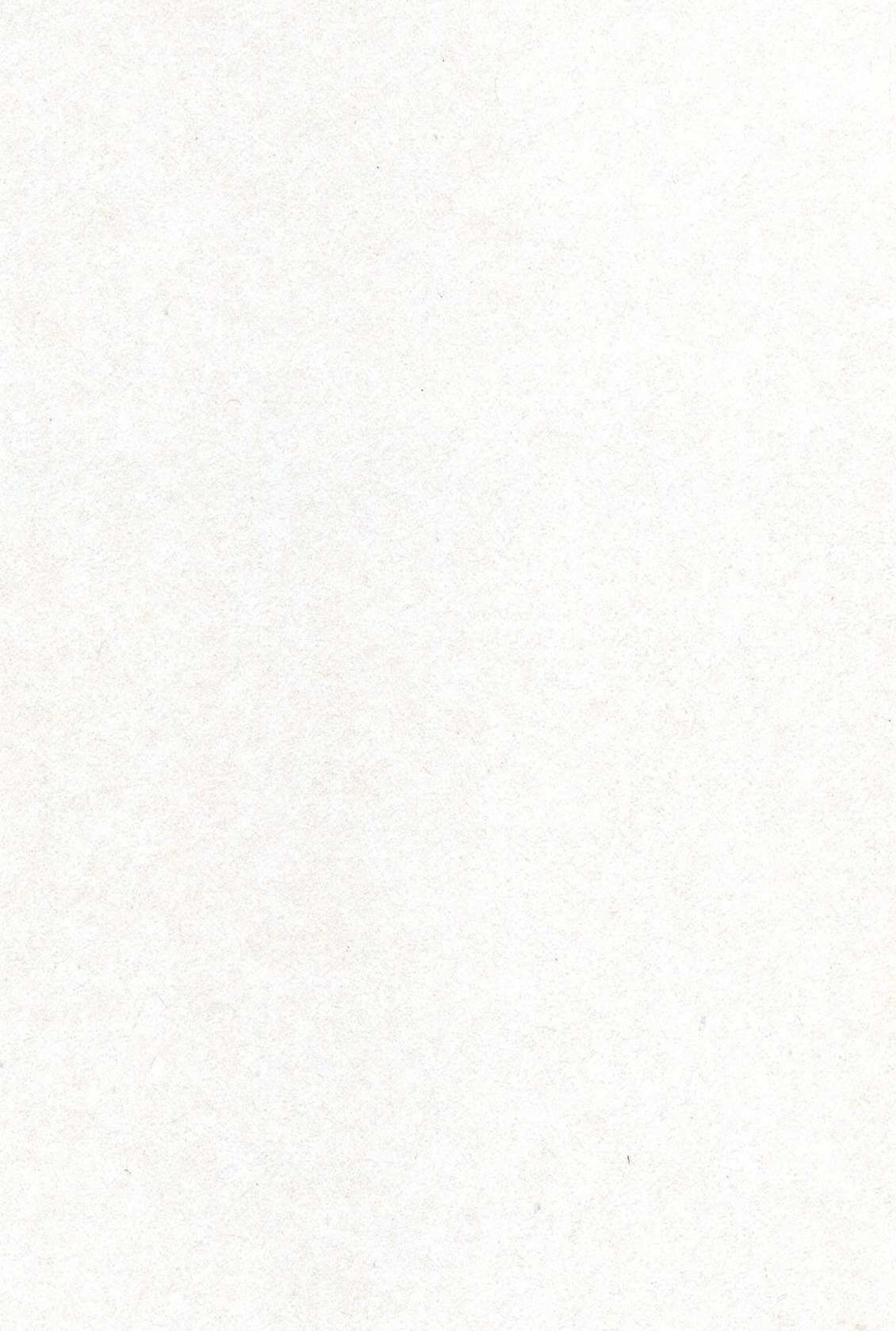
Sigamos, pues, la pista a Alan Turing. Muchos lo consideran el padre de los ordenadores actuales, es decir, de ordenadores completos, con un programa introducido que se ocupa de que la máquina realice la tarea correspondiente. Nuestro investigador trabajó para el Departamento de Comunicaciones del Ministerio de Asuntos Exteriores británico (léase centro de criptoanálisis británico), en los años de la Segunda Guerra Mundial, diseñando y fabricando toda una familia de ordenadores: los Robinson. El primero, Heath Robinson, comenzó a funcionar en 1940, y más tarde le siguieron los restantes elementos de la familia: "Peter Robinson", "Robinson and Cleaver" y "Super Robinson" (el nombre de la familia lo tomaron de unos tebeos famosos en la época). En esta serie se utilizaba una cinta de formato fijo y otra para los datos, leyéndose una contra la otra. El sistema de almacenamiento era electrónico.

Sabía usted que...

¿Sabía que el ASCC, máquina antecesora del primer ordenador, el ENIAC, fue financiado por la IBM (que, por el contrario, estuvo fuera del proyecto posterior, el ENIAC)? El diseño se debió al profesor Howard Aiken, de la Universidad de Harward, a la que fue finalmente donada la máquina. Esta medía 15 metros de longitud por 24 de anchura. Contenía unos 760.000 elementos, conectados por 800 km. de cables. Utilizaba relés, y podía tomar algunas decisiones restringidas, por lo que era prácticamente un ordenador. Fue una máquina robusta, que estuvo conectada ininterrumpidamente en la Universidad durante muchos años.

Pero todavía no hemos alcanzado el final de esta historia. La Guerra Mundial supuso un enorme avance tecnológico. Nada se detenía, todo era mejorable, y en aquellos años apareció también otra serie de máquinas, mucho más avanzadas que las Robinson: las Colossi, desarrollada por el profesor M. H. A. Newman. Estas máquinas estaban limitadas generalmente a una tarea específica, como casi todas las máquinas existentes en esas fechas.

A partir del momento en el que terminó la guerra, la mayor parte de los científicos que participaron en los proyectos se diseminaron, creando muchos de ellos sus propios equipos.



Sistemas expertos



UEDEN construirse sistemas informáticos que sean realmente "expertos" en alguna materia?

Bajo el nombre de "sistemas expertos" se agrupa un conjunto de aplicaciones informáticas que incorporan técnicas de Inteligencia Artificial (IA) y que puede estar desarrollado con procedimientos y concepciones realmente muy diversas. De hecho, de entre las aplicaciones prácticas de todos los conocimientos científicos incluidos en lo que se ha dado en llamar IA los "sistemas expertos" (SE) es la más conocida y, probablemente, la más extendida.

Con un SE se trata de "sustituir" o "complementar" a los "expertos humanos" para la realización de una serie de tareas muy diferentes: análisis de datos o informaciones, toma de decisiones o sugerencia de acciones a tomar (basadas o no en los análisis antes indicados), monitorización de funcionamiento de equipos o instalaciones, simulación y estudio de situaciones complejas, formación, etc.

Conviene indicar cuanto antes que las capacidades de "razonamiento" o inferencia de cualquier sistema informático son, desde luego, inferiores a las humanas y, además, conceptos como "imaginación" o "intuición" pueden ser difícilmente aplicables (o sólo se pueden aplicar con muchas restricciones) a un programa o conjunto de programas que funcionen en un ordenador. Sin embargo, con equipos adecuados y con técnicas especiales (como sucede en estos SE) se pueden obtener cualidades por encima de las humanas en algunas facetas: en capacidad de almacenamiento, en rapidez, en seguridad...

En efecto, en un SE se utiliza una cantidad de información (de "conocimientos", se suele decir) que una persona difícilmente podrá considerar "simultáneamente" en su proceso mental (aunque tanto el ordenador como la persona los utilicen sucesivamente); por otro lado, la rapidez de acceso a las informaciones almacenadas y la rapidez de proceso son mayores en el ordenador; además, la seguridad en el uso de estos conocimientos es, sin duda, superior en el SE, sin que ningún dato o información sea olvidado en un proceso de inferencia y de tal modo que no intervienen en la actividad ni el estado de ánimo, ni el cansancio, ni ninguna otra condición subjetiva.

De todos modos, todas estas cualidades o ventajas descritas (y otras que se podrían indicar) son comunes (al menos en parte) a todos los sistemas, aplicaciones o programas informáticos. Hay, sin embargo, algunas características que diferencian a los SE de los sistemas informáticos tradicionales: básicamente un SE es un sistema informático "basado en el conocimiento" más que "basado en los datos" (veremos más adelante qué significa esto); además, hay otros elementos diferenciales en la propia estructura del sistema, en las "herramientas" informáticas que se utilizan y en las "utilidades" que se pueden obtener de él.

Intentemos aclarar el punto clave: un SE es un sistema "basado en el conocimiento" (y de hecho éste es otro nombre con el que se suele conocer a los SE). En un programa o sistema informático convencional el (los) algoritmo(s) de proceso permanecen inalterados, con ciertas salvedades (es decir, el "progra-

ma" está definido) y se opera sobre una base o conjunto de datos generalmente variable. Por el contrario, en los SE, se separan en dos conjuntos de elementos las informaciones de que se dispone respecto al proceso a realizar: por un lado se define un proceso de deducción o inferencia que permanece fijo (se llama "motor de inferencia") y por otro se almacena una colección de reglas ("base de conocimientos") donde se contiene la información que está disponible sobre el tema en que deseamos que sea experto el sistema. El "motor de inferencia" opera sobre la "base de conocimientos" para obtener nuevas informaciones a partir de los datos de entrada.

Es decir, en un sistema experto aparecen tres elementos básicos: una base de conocimientos (formada a partir de la información que el experto humano ha "transmitido" al sistema), un sistema de obtener deducciones o sistema de proceso de los conocimientos contenidos en la base de conocimientos, y una base de datos con los que va a operar el conjunto anterior. Normalmente el sistema de deducción o inferencia ("motor de inferencia") es fijo y ha sido definido en función del tipo de problema a resolver, pero la base de conocimientos se va actualizando con el funcionamiento del sistema.

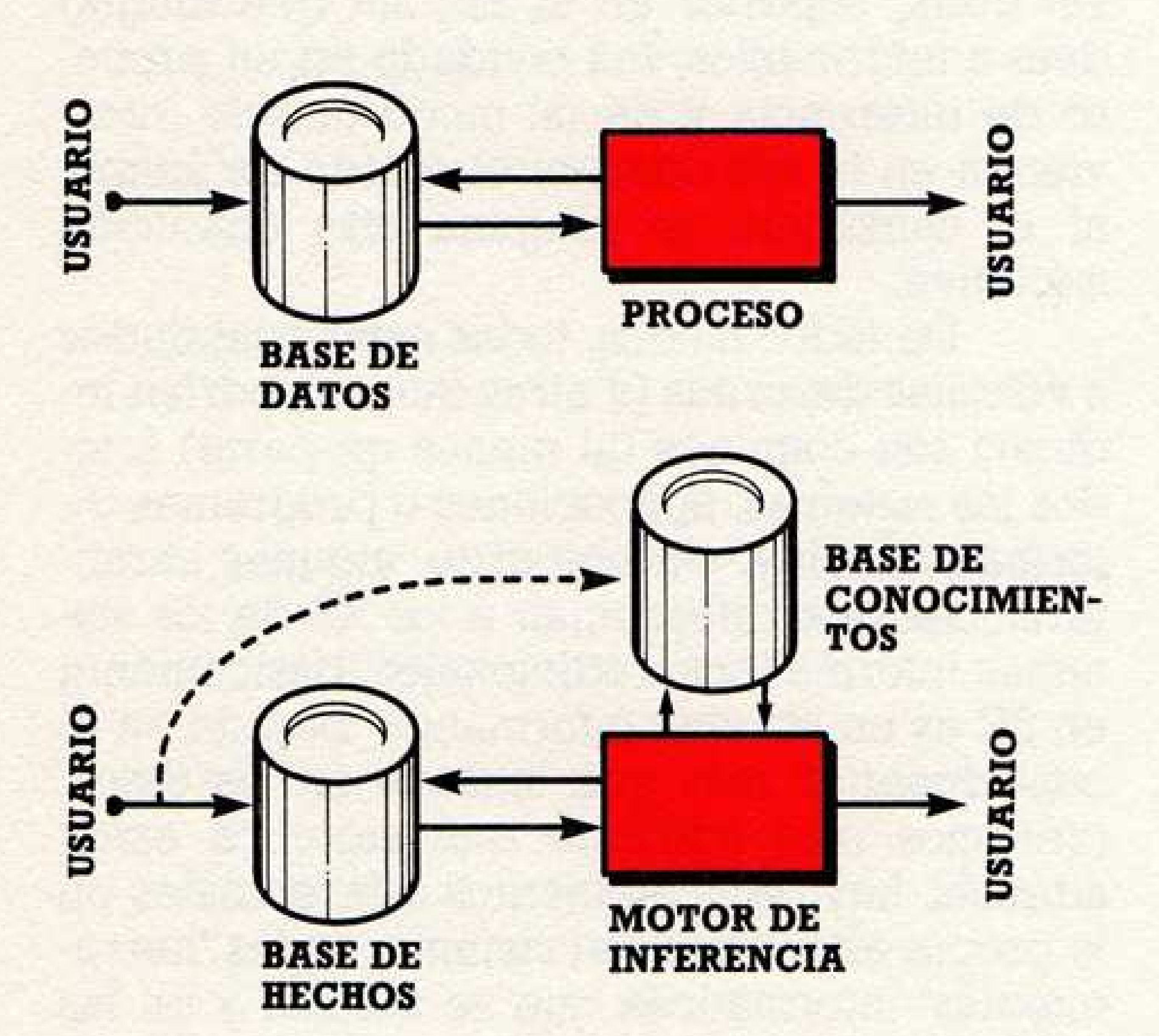


Fig. 1.—Modificación de la estructura de procesamiento en un sistema basado en el conocimiento.

Otro elemento diferencial en los sistemas expertos suele ser el conjunto de "herramientas" informáticas con el que ha sido desarrollado: en efecto, el auge de este tipo de aplicaciones ha venido de la mano de un enorme desarrollo paralelo experimentado por los lenguajes de programación específicamente pensados para la IA (LISP, especialmente adecuado para el proceso de listas; PROLOG, diseñado para el proceso de expresiones lógicas; POPLOG, combinación de los anteriores, etcétera) y los esfuerzos realizados por todos los Centros de Investigación y las Compañías para la creación de numerosos "sistemas-concha" (Shell): estos sistemas o entornos de programación y/o desarrollo, son conjuntos de "herramientas" software concebidas para facilitar la tarea de quien va a desarrollar un sistema experto; se incluyen en los sistemas-concha de aplicaciones en IA, los diversos lenguajes disponibles o dialectos de ellos (INTERLISP, MCCLIPS, ...TURBOPROLOG...) junto con editores, sistemas de representación del conocimiento, manejadores de entornos gráficos, sistemas de seguimiento de trazas, etc. Ejemplos de estos sistemas son KEE, LOOPS, OPS5, etc.

Por otro lado, existen también disponibles (para su utilización en la creación de sistemas expertos) los llamados "sistemas esqueleto". Si de un sistema experto extraemos la información específica existente sobre un área concreta (base de conocimientos) y dejamos el motor de inferencia, los elementos de representación del conocimiento, los módulos de interface con el usuario, etc., se puede tener un "sistema experto vacío" que puede ser utilizado como una herramienta de propósito general para construir un sistema experto "sin más que" incluir conocimientos específicos sobre cualquier otro tema.

Por último, suelen disponer los sistemas expertos de dos módulos (o grupos de módulos) de programación adicionales que incrementan enormemente su utilidad: la interfaz con el usuario y el subsistema de explicaciones.

La interfaz de usuario es el subsistema que "dialoga" con el usuario para la obtención (en un entorno lo más cómodo y grato posible) de la información necesaria para el sistema (obtención de reglas para la base de conocimientos y obtención posterior de los hechos sobre los que va a trabajar el sistema). Para ayudar en estas tareas se están desarrollando numerosas experiencias de interpretación de lenguaje natural, de acceso "inteligente" a bases de datos, etc.

El subsistema de explicaciones o traza de un SE es la parte de él dedicada a justificar las recomendaciones que el SE emite o las decisiones que toma, y a explicar el camino seguido para llegar a dichas conclusiones. En el fondo es (aparte los detalles más técnicos) un sistema de registro del camino que el motor de inferencia ha seguido para llegar desde los datos de partida ("hechos") hasta la conclusión obtenida. Esta cualidad de "justificar" las decisiones tomadas es, en numerosas aplicaciones, una de las claves de éxito de los SE.

Sistemas expertos clásicos y su utilidad

CASNET (1977). Diagnóstico de glaucoma.

CRYSAUS. Estudio de la cristalografía de las proteínas. A partir de diversas fuentes de conocimiento perfecciona la interpretación de los "mapas de densidad de electrones".

DENDRAL (1965). Investigación de la estructura orgánica molecular de compuestos desconocidos a partir de las restricciones deducidas de los datos aportados.

EXPERT. Diseñado para ayuda de los investigadores. Les dirige en el diseño y prueba de modelos de consulta.

GUIDON. Sistema complementario de MYCIN diseñado para que los estudiantes usen dicho SE de un modo eficaz. Utiliza un diálogo con el estudiante para guiarlo en el uso de MYCIN.

HERSAY. Interpretación de frases en función de los datos aportados por el contexto.

INTERNIST (1975). Relaciones entre los descriptores de enfermedades y diagnóstico en medicina interna.

MACSYMA (1971). Realización de operaciones de cálculo integral y diferencial por búsqueda del camino óptimo entre el conjunto de operaciones básicas propuestas por los expertos.

MYCIN (1975). Diagnóstico de enfermedades infecciosas a partir de los síntomas dados o deducidos.

PROSPECTOR (1976). Localización de yacimientos minerales partiendo de la configuración y características del terreno.

TEIRESIAS. Introducción y modificación del conocimiento del sistema MYCIN; detecta fallos en las cadenas de razonamiento, perfeccionando, en consecuencia, la base de conocimientos.

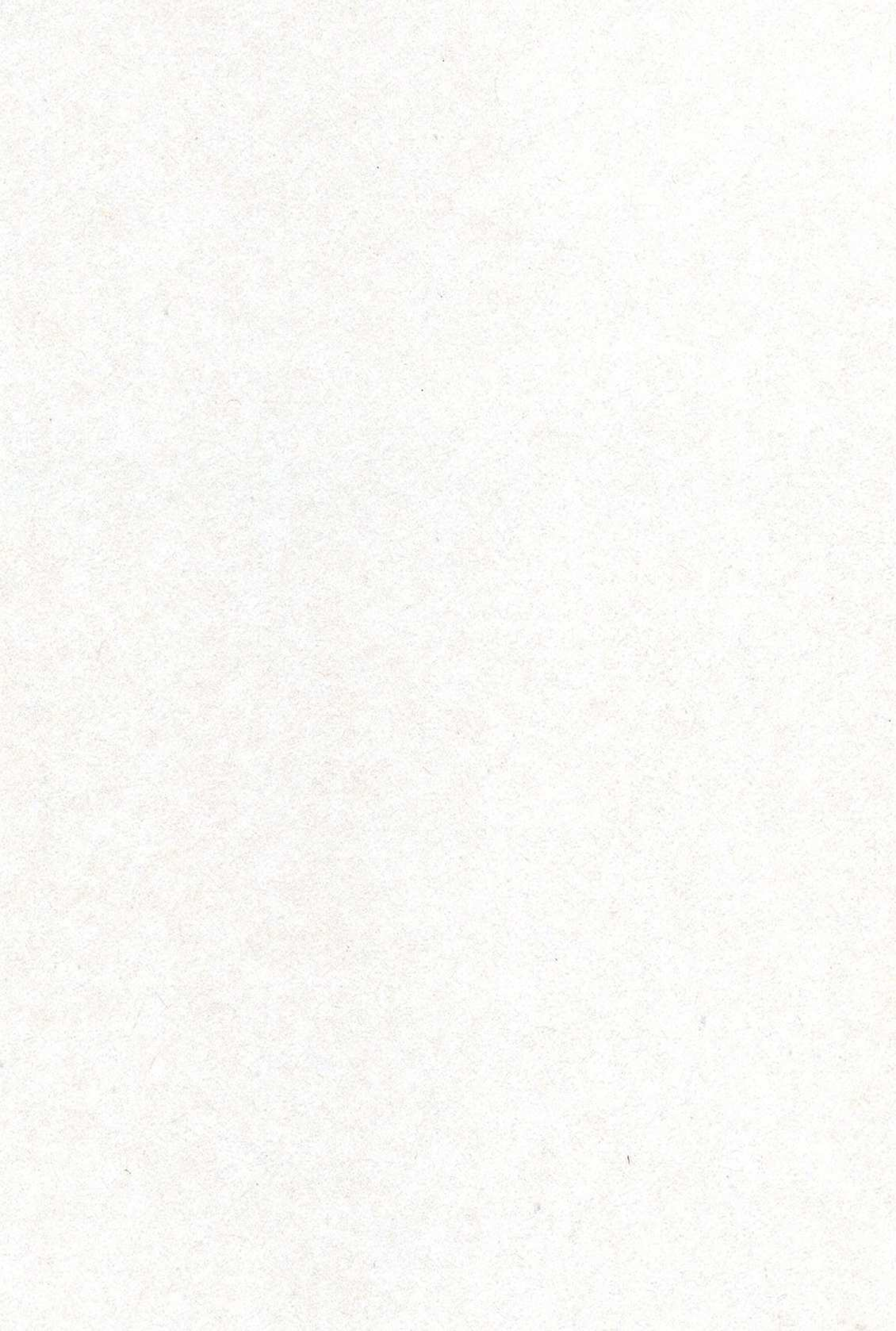
Areas de aplicación de los sistemas expertos

A) Areas "más prometedoras" (según Johnson —1984—) para la aplicación de sistemas expertos:

- ordenadores, electrónica y comunicaciones: diagnóstico de fallos y mantenimiento; ordenadores "front-end" expertos; diseño y configuración de sistemas.
- extracción y exploración de petróleo: control de extracción, análisis de datos sísmicos.
- servicios financieros: soporte en ventas y gestión; gestión de cartera.
- aplicaciones militares: mantenimiento y diagnóstico de averías; entrenamiento y formación; evaluación de informaciones tácticas.

B) Areas de desarrollo de sistemas expertos dentro del Proyecto de 5.ª Generación japonés o en actividades relacionadas con él (Ishizuka, 1984).

- Diagnóstico y consulta médica.
- Supervisión y control de plantas industriales.
- Análisis de tasa de fallos estructurales.
- Gestión empresarial y ofimática.
- Diseño asistido por ordenador.
- Proceso de imágenes.
- Acceso "inteligente" a Bases de Datos.



TERMINOLOGIA

GLOSARIO DE TERMINOS UTILIZADOS EN SISTEMAS EXPERTOS

Quinta generación. Ordenadores desarrollados basándose en técnicas de Inteligencia Artificial y que utilizan dichas técnicas en sus procesos internos. Se programan en lenguajes también llamados de 5.ª generación (PROLOG, básicamente).

Algoritmo. Proceso o conjunto de reglas definido para resolver un problema o realizar una tarea específica. Normalmente puede ser representado mediante una(s) expresión(es) matemáticas(s) a la(s) que también se llama algoritmo.

CAD. Computer-aided Design. Diseño asistido por ordenador. Aplicación o sistema que ayuda en el diseño ofreciendo facilidades gráficas, diagramas predefinidos, etc.

CAI. Computer-aided Instruction. Enseñanza asistida por ordenador. Programa(s) que proporcionan un curso de enseñanza autónoma basada en el ordenador.

CAL. Computer-aided Learning. Aprendizaje asistido por ordenador. Para algunos sinónimo de CAI, pero normalmente referido, más bien, a la enseñanza reglada o académica.

CAM. Computer-aided Manufacturing. Fabricación asistida por computador. Aplicación que cubre todas las etapas del proceso de fabricación. Muy normalmente relacionada con un sistema de CAD.

CAT. Computer-aided Training. Instrucción asistida por ordenador. Parecida a CAI, pero para actividades profesionales y de aprendizaje de manejo o control de instrumentación, plantas industriales, etc.

Conocimiento. Información(es) estructurada para su utilización en un sistema "inteligente" (normalmente un SE).

—, Base de. Conjunto de informaciones sobre las que opera el sistema experto. La creación de la base de C es uno de los problemas costosos de la implantación de un S.E; El conjunto de información que proporciona el experto humano se estructura en la Base de conocimiento del S. E. Se suelen considerar dos tipos de conocimientos en la Base: hechos ("Amaya es hija de José") y relaciones ("El marido de la hija de alguien es su yerno").

—, Ingeniería del. Técnicas de realización y manejo de S. E. La persona que conoce y utiliza estas técnicas es llamada, por tanto, ingeniero del Conocimiento. Esta persona es la que diseña el S. E. según las indicaciones del experto humano y gestiona la creación de la Base de Conocimientos a partir de la experiencia de dicho experto humano.

—, Sistema de representación del. Técnica utilizada en la formalización de las informaciones incluidas en la Base de Conoc. Los procedimientos más usuales de representar el conocimiento son:

— Lógica de primer orden.
— Redes semánticas.
— Marcos de representación ("Frames" en inglés).
— Reglas de producción.

Conchas, (sistemas "shell"). Conjunto de "herramientas" de software necesarias para construir un sistema experto: suelen incluir lenguajes para la construcción de la base de concimiento, algún monitor de inferencia

genérico. Algún sistema de interrogación respuesta para acceder al conocimiento y otros módulos auxiliares: de edición, de preparación de datos, de traza del sistema, etc.

Decisión Support System. Sistema de Soporte en la toma de decisiones. Básicamente un sistema experto preparado para este tipo de actividad.

Dominio (de conocimiento). Area especializada cuyas informaciones son susceptibles de ser formalizadas para su introducción en un sistema experto.

Encadenamiento. Procedimiento de obtención de conclusiones en un sistema experto. El encadenamiento hacia adelante (Forward chaining) consiste en partir del grado de evidencia de los datos originales para deducir de ello la probabilidad de los resultados a obtener y seleccionar estos resultados.

En el encadenamiento hacia atrás, la operación se realiza a la inversa partiendo del resultado a obtener para decidir qué regla hay que aplicar y, en consecuencia, de qué datos hay que partir.

Normalmente un S. E. utiliza uno de estos dos procedimientos, pero hay algunos que usan ambas técnicas.

Fuzzy Logic. Lógica difusa. Técnica de razonamiento teniendo en cuenta informaciones no ciertas o información parcial.

Front end, procesador. Ordenador especializado en el manejo de comunicaciones. En las grandes instalaciones informáticas el front-end gobierna los terminales y sirve de intermediario entre la red y el ordenador central, liberando a éste de soportar los protocolos de comunicaciones y el "diálogo" con las estaciones periféricas.

Heurístico. Método de resolver problemas mediante ensayos de tipo "prueba-y-error" (se intenta un procedimiento y se ve cuán cerca está de la solución, para modificar el ensayo y volver a probar), en contraposición a la utilización de un algoritmo predeterminado.

Inferencia. Deducción. Proceso de obtener un resultado válido a partir de las informaciones de que se dispone.

Inferencia, Motor de. Módulo(s) del sistema experto encargado(s) de obtener conclusiones a partir de los datos originales ("hechos") y teniendo en cuenta las informaciones contenidas en la Base de Conocimientos.

Lenguaje natural. Lenguaje en que nos expresamos las personas (como el castellano, catalán, vasco, etc.). En contraposición a los lenguajes formales de programación, etc.

—, No-procedural. Lenguaje de programación concebido más bien para el proceso de aplicación en vez de para la definición de procedimientos algorítmicos, como el resto de los lenguajes.

Lista. Conjunto ordenado de palabras (llamadas "átomos"). En lenguaje LISP, este conjunto debe ir entre paréntesis y constituye una especie de frase básica de dicho lenguaje.

Motor de inferencia. Ver **Inferencia, motor.**

Reglas de producción. Expresión del conocimiento formulado en forma de SI (condición), ENTONCES (resultado). Es uno de los procedimientos de representación de la información del experto en una Base de Conocimientos.

Set-down knowledge. Conocimiento disponible ya en una forma muy válida para su inclusión en la Base de Conocimientos. (Por ejemplo: manuales, procedimientos legales, etc.).

Top-down, design. Técnica de diseño que parte de los conceptos u objetivo generales y los subdivide en otros de menor nivel y éstos a su vez en otros, dentro de un esquema estructurado.

Traza. O explicación. Relación de pasos que ha seguido el motor de inferencia (a través de la Base de Conocimiento) para obtener la respuesta que se ha dado al usuario.

VOCABULARIO DE INFORMATICA

Almacenamiento intermedio. También llamado almacenamiento temporal de E/S. Zona de memoria destinada al almacenamiento temporal de datos antes de su envío (recepción) hacia (o desde) un dispositivo periférico.

Almacenamiento masivo. Este término se utiliza para designar a los dispositivos de almacenamiento externo de gran capacidad.

Alta resolución. Modo de representación de gráficos que utiliza gran número de pixels o unidades de representación gráfica. También se refiere a las pantallas que son capaces de actuar en este modo de representación.

ALU (Arithmetic and Logic Unit). Unidad Aritmética y Lógica. Parte de la CPU donde se realizan las operaciones aritméticas y lógicas.

Analógico. Proceso, dispositivo o señal que trabaja con (o representa) datos en forma continua, en contraposición a lo que sucede en los procesos digitales.

Analógico/digital, conversor. Dispositivo que proporciona una señal digital (es decir, que toma valores discretos dentro de una gama) aparte de la señal analógica (continua) de entrada.

Anchura de banda. Gama de frecuencias que pueden ser transmitidas o procesadas por un dispositivo de comunicaciones.

Anidar. Incluir un bucle de proceso dentro de otro bucle.

Anillo, red en. Red de comunicaciones en la cual los elementos se conectan en serie; es decir, la información pasa de uno a otro.

ANSI. Acrónimo de American National Standard Institute, oficina de la Administración de los EE.UU. que elabora disposiciones para la normalización.

APL. Lenguaje de programación de alto nivel desarrollado fundamentalmente con fines matemáticos. Utiliza multitud de operaciones, y el usuario puede definir sus propios operadores. Generalmente los programas APL resultan muy concisos.

Arbol. Estructura de datos articulada en diferentes niveles relacionados entre sí de un modo jerárquico a partir de un único elemento (nodo inicial o raíz).
Grafo representativo de esta estructura.

Arbol binario. Arbol en el cual de cada elemento (nodo, en el grafo) derivan uno o dos nuevos elementos (ramas).

Archivo. También llamado fichero. Conjunto de registros organizado de una manera estructurada para ser utilizado por una o varias aplicaciones.

Area intermedia de la memoria. Zona de la memoria utilizada para introducir datos de forma provisional. Normalmente se utili-

za para eliminar los problemas debidos a las distintas velocidades de los dispositivos de tratamiento de datos, y que las transferencias se realicen adecuadamente.

Argumento. Parámetro. Valor que se pasa a un procedimiento o a una función en el momento de llamarla.

Arquitectura del ordenador. Organización general y especificaciones del sistema del ordenador. Toma en consideración todos los elementos del equipo: componentes, conexiones, organización general, control, etc.

Arrastre hacia adelante. Se utiliza en sumadores paralelos de multibits. El elemento individual del sumador inmediatamente anterior es capaz de detectar el momento en el que va a producirse un arrastre como resultado de la suma de los siguientes bits menos significativos de los sumandos.

Array (ver matriz).

ASCII. Siglas de American Standard Code for Information Interchange. Sistema de codificación de caracteres muy utilizado. El código utiliza 7 bits, sin bit de paridad, y, por consiguiente, dispone de 128 signos diferentes.

Asíncrono. Se refiere a unas transmisión de datos que comienza con la recepción de una señal que indica que la operación anterior ha finalizado. Permite, pues, que existan retardos importantes entre dos señales.

Asíncrono, ordenador. Ordenador en el cual cada operación comienza a recibir una señal generada como resultado de la terminación de la operación anterior, o después de comprobar la viabilidad de la operación que se va a realizar. En contraposición con un ordenador síncrono.

Asignar. En los lenguajes de programación, sentencia por la cual se da un valor a una variable. También puede aplicarse a la memoria, asignando una dirección fija. Si la asignación es dinámica, puede cambiar en el transcurso del programa.

Assembler (ver ensamblador).

Atenuación. Pérdida de intensidad de una señal durante la transmisión.

Atomo. Elemento mínimo de información considerado en algunos sistemas de representación del conocimiento.

Autocarga. Carga automática. Carga inicial automática del programa. Una señal sin calificar del ordenador base produce que los datos se lean y se transfieran al ordenador.

Autoejecutable. Programa que arranca su ejecución de modo automático.

Autómata. Dispositivo de evaluación automática para análisis de las palabras (cadenas) de un lenguaje formal y decisión de su inclusión en él, o para producción de una nueva cadena.

Máquina física que realiza acciones sofisticadas "parecidas" a las humanas: robot.

Autómata finito. Tipo de autómata sencillo. Su estado cambia con cada símbolo de la cadena de entrada en primer lugar, y del estado corriente en segundo lugar. Se llama finito porque el conjunto de todos sus posibles estados debe ser un número finito.

Automática, vuelta del carro. Mecanismo de control de máquinas de escribir y otros dispositivos que controla automáticamente el salto de línea, espaciados, y el salto de página del papel o del impreso.

Autóctono (ver stand-alone).

Auxiliary storage (ver almacenamiento, memoria auxiliar de).

Axioma. Aseveración básica de un sistema lógico que se introduce sin demostración.

